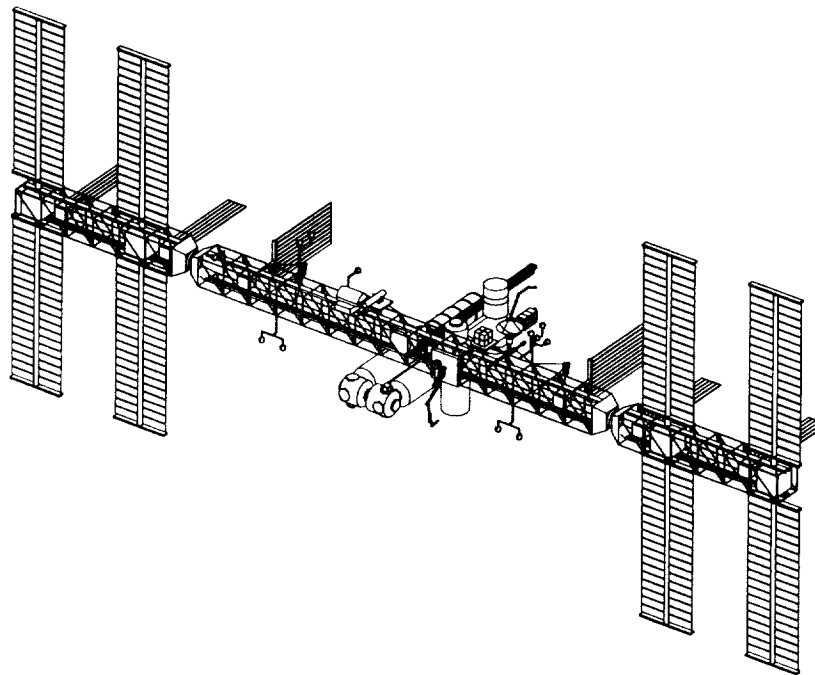


A Review of Space Station Freedom Program Capabilities for the Development and Application of Advanced Automation



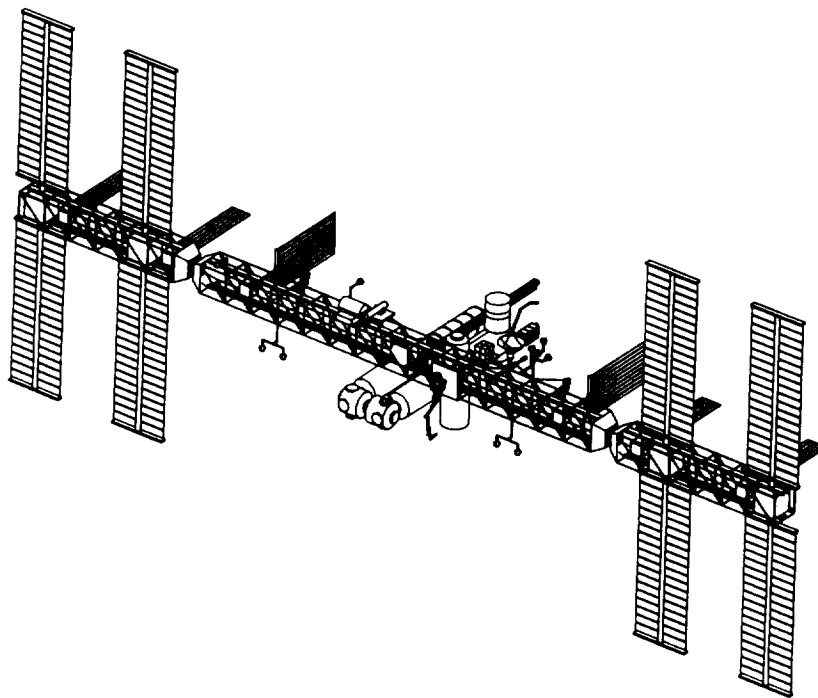
(NASA-CR-188252) A REVIEW OF SPACE
STATION FREEDOM PROGRAM
CAPABILITIES FOR THE DEVELOPMENT
AND APPLICATION OF ADVANCED
AUTOMATION (Mitre Corp.) 128 p

N94-70054

Unclas

Z9/18 0190876

A Review of Space Station Freedom Program Capabilities for the Development and Application of Advanced Automation



MITRE

A Review of Space Station Freedom Program Capabilities for the Development and Application of Advanced Automation

Steven E. Bayer
Richard A. Harris
Lois W. Morgan
James F. Spitzer

December 1988

MTR-88D00059

SPONSOR:
NASA/JSC
CONTRACT NO.:
NAS9-18057
PROJECT:
3100V
DEPT:
W-124

This document was prepared for authorized distribution.
It has not been approved for public release.

The MITRE Corporation
Civil Systems Division
7525 Colshire Drive
McLean, Virginia 22102-3481

Department Approval: Stuart Bell for
James F. Spitzer

MITRE Project Approval: Stuart Bell for
Edwin S. Herndon

Peer Reviewer: Robert M. Myers
Robert M. Myers

ABSTRACT

The use of advanced automation and robotics within the Space Station Freedom Program is motivated by potential benefits, such as reduced life cycle costs and technology transfer to industry. As advanced automation and robotics technologies mature, they will be utilized within systems onboard Space Station Freedom and the space platforms, as well as within control centers on the ground. The National Aeronautics and Space Administration has substantial expertise related to the development and testing of conventional software systems for in-flight use; however, little experience or knowledge exists with respect to engineering, testing, and certifying systems which utilize advanced automation technologies. These technologies will need to be elevated to a high level of readiness before they can be safely used in space.

This document reviews, for Space Station Freedom Program managers, Space Station Freedom Program advanced automation capabilities and plans, including design and research facilities (i.e., laboratories and test beds), operational and support facilities (i.e., Software Support Environment and Multi-System Integration Facility), and existing prototypes. In addition, preliminary concepts and strategies are presented for the evolution of Space Station Freedom Program facilities in support of the development and testing of advanced automation.

Suggested Keywords: advanced automation, test beds, expert systems, knowledge-based systems, artificial intelligence, flight certification, verification and validation, Space Station Freedom, Space Station Freedom Program

EXECUTIVE SUMMARY

INTRODUCTION

The use of advanced automation and robotics within the Space Station Freedom Program (SSFP) is motivated by potential benefits, such as reduced life cycle costs and technology transfer to industry. As advanced automation and robotics technologies mature, they will be utilized within systems onboard Space Station Freedom and the space platforms, as well as within control centers on the ground. The National Aeronautics and Space Administration (NASA) has substantial expertise related to the development and testing of conventional software systems for in-flight use; however, little experience or knowledge exists with respect to engineering, testing, and certifying systems which utilize advanced automation technologies. These technologies will need to be elevated to a high level of readiness before they can be safely used in space.

The purpose of this task is to review, for SSFP managers, SSFP automation capabilities and plans, including design and research facilities (i.e., laboratories and test beds), operational and support facilities (i.e., Software Support Environment and Multi-System Integration Facility), and existing prototypes. In addition, preliminary concepts and strategies are presented for the evolution of SSFP facilities in support of the development and testing of advanced automation. This review will provide a base upon which subsequent evolution plans, relative to the coordinated use of advanced automation within the SSFP, can be generated for SSFP flight systems and ground facilities.

This task is funded by the Space Station Freedom Advanced Development Program, Strategic Plans and Programs Division (Code ST), Office of Space Station (OSS). This task is managed by the Intelligent Systems Branch of the Engineering Directorate at the Johnson Space Center (JSC).

For the purposes of this task, advanced automation is defined as the use of concepts and methods of high-level symbolic inference by a computer and the symbolic representation of the knowledge to be used in making inferences so as to make a machine behave in ways that humans recognize as intelligent behavior. The technical scope of this study is restricted to advanced automation only and, therefore, excludes robotics.

Several studies that preceded this task provide a firm foundation for the definition of advanced automation software engineering facilities for the SSFP. The following studies are especially important: ongoing reports of the NASA Advanced Technology Advisory Committee (ATAC); the Systems Autonomy Technology Program (SATP); the Space Station Technology Development Mission Requirements Definition for Advanced Automation Study; and the Space Station Advanced Automation Study Final Report.

The MITRE Corporation's support of Space Station Freedom development has included support for the development of requirements for many SSFP ground and space systems. To extend this broad knowledge base in support of this task, both NASA and non-NASA advanced automation test beds and laboratories were visited.

This summary briefly discusses design and research facilities (laboratories and test beds), operational and support capabilities (the Software Support Environment and the Multi-System Integration Facility), and evolution paths for advanced automation.

DESIGN AND RESEARCH FACILITIES

Design and research capabilities of the SSFP include existing SSFP laboratories and test beds for on-board and ground systems. The laboratories and test beds, major test bed integration activities, and other significant research activities discussed in the body of this document are briefly introduced in the following paragraphs.

Laboratories and Test Beds

Data Management System Test Beds. The Data Management System (DMS) will provide hardware resources and software services on the manned base that support the data processing and communications requirements of Space Station Freedom systems and flight elements. There are two DMS test beds: the JSC DMS Test Bed, oriented toward functions and services; and the International Business Machines (IBM) DMS Test Bed, oriented toward hardware design testing. Within this document, the phrase *DMS Test Bed* refers to the JSC DMS Test Bed.

JSC Operations Management System Prototypes. The DMS Test Bed is the host facility for the JSC Operations Management System Prototypes. These prototypes target activities such as the commanding, monitoring, and control of flight systems which can benefit from increased automation. These operations management tasks have historically been performed by the flight crew, ground controllers, and engineering support personnel. Space Station Freedom operations management concepts will be implemented as the Operations Management System (OMS), a major application of the DMS software. Advanced automation technologies, as they mature, will be utilized to provide increased autonomy.

Two prototypes, developed by The MITRE Corporation, are currently being used within the DMS Test Bed. The Integrated Status Assessment expert system prototype performs station-wide failure diagnosis. The Procedures Interpreter prototype demonstrates a possible implementation of execution of the short term plan, one of the OMS functions.

Thermal Control System Test Beds. The Thermal Control System will maintain the Space Station Freedom equipment and customer payloads within their allowable operational and non-operational temperature range. The JSC Thermal Test Bed is a complete thermal system composed of test articles (i.e., pumps, radiators, evaporators, condensers, central thermal buss). Several sets of thermal test articles are being tested and compared. The physical thermal test bed at Ames Research Center (ARC) is a small brassboard of one configuration of the JSC Thermal Test Bed. ARC is responsible, in association with JSC thermal engineers, for the development of Thermal Expert System (TEXSYS). TEXSYS will control and monitor one configuration of the thermal test bed and will participate in the 1988 Systems Autonomy Demonstration Project (SADP) single-system demonstration, as well as later SADP demonstrations.

Electrical Power System Test Beds. The Electrical Power System (EPS) will provide, distribute, and store electrical power for the Space Station Freedom systems and elements. The Lewis Research Center (LeRC) is responsible for the development of the EPS, which is the part of the power system which generates the electrical power. Marshall Space Flight Center (MSFC) is responsible for developing the Space Station Module (SSM)/Power Management and Distribution (PMAD) System, which is the part of the power system which distributes the power within a Space Station Freedom module. The JSC Generic Electrical Power Distribution and Control Test Bed provides a simulation of the Space Station Freedom power generation and distribution system to evaluate candidate solutions to the Space Station Freedom power system in support of the ETC integration efforts.

JSC Guidance, Navigation, and Control Emulator Test Bed. The Guidance, Navigation, & Control (GN&C) Emulator Test Bed allows the testing of GN&C design options and operations

concepts. The test bed consists of a set of small computers connected with communication busses, emulating the distributed architecture, software, programs, and other functions of the GN&C System.

JSC Communications and Tracking System Test Bed. The Communications and Tracking (C&T) System provides all Space Station Freedom manned base communications services, including audio, video, space-to-space communications, and space-to-ground communications. The C&T system also provides the necessary tracking services to GN&C. The Communications and Tracking Control and Monitoring (C&M) Subsystem Test Bed is used to develop and evaluate candidate software for the Space Station Freedom C&T system.

ARC Advanced Architecture Test Bed. The goals of the Advanced Architecture Test Bed are to investigate the hardware and software issues relative to the application of onboard multi-processor systems to NASA missions and to provide the required prototyping capability for transfer of the architecture technologies to specific NASA projects. Example technical problems addressed by this test bed include real-time fault tolerant architectures, management and control of large distributed knowledge bases, and operating systems for a distributed heterogeneous environment. A major purpose of the Ames test bed is to investigate advanced computing concepts and to feed the results to JSC for possible use within the DMS Test Bed.

Other DMS Nodes. The Human Computer Interaction Node of the JSC DMS Test Bed is a prototype configuration which will allow the exploration of human factors issues through the presentation of information produced by the OMS and core systems nodes. The MPAC Displays and Control (D&C) Node is used for the prototyping of the fixed MPAC hardware configuration and on-board crew controlling and monitoring activities in a Space Station Freedom environment.

Advanced Automation Activities. Many of these laboratories and test beds are developing advanced automation prototypes for their respective systems. The following list shows the prototypes being developed or tested at the laboratories and test beds discussed in this study.

- | | |
|------------------------------|---|
| • JSC OMS Test Node | Integrated Status Assessment expert system prototype
Procedures Interpreter prototype |
| • ARC OAST Laboratory | Thermal Expert System (TEXSYS) |
| • LeRC EPS Test Bed | Power Management and Distribution System (PMACS)
Automated Power Expert (APEX) |
| • MSFC SSM/PMAD Test Bed | Loads Priority List Management System (LPLMS)
Loads Enable Scheduler (LES)
Fault Recovery and Management Expert System (FRAMES) |
| • JSC GN&C Emulator Test Bed | On Board Check Out (OBCO) |
| • JSC C&T C&M Test Bed | Local Controller Fault Manager expert system
Central Processor Resource Manager expert system |

Most of these prototypes (e.g., TEXSYS, APEX) are diagnostic systems, the most mature application of expert systems technology. A wide variety of development platforms are being used, which is appropriate for early development of the Space Station Freedom. As the Station Freedom design evolves and matures, and technology demonstrations take place, these laboratory and test bed activities should evolve toward high-fidelity, standard, flight-like hardware and software platforms. In addition, interface issues between advanced automation technologies and conventional software engineering tools

(e.g., Ada) should be addressed. These research and development activities are discussed in more detail within the body of the paper.

Major Test Bed Integration Activities

End-to-End Test Capability. The Space Station Information System (SSIS) comprises the information processing and communications capabilities that will be involved in handling operational and scientific data generated or used within the SSFP. The SSIS will provide end-to-end connection of, and a variety of information services to, a diverse and geographically distributed user community. This end-to-end connectivity will extend from the flight systems and payloads, to the ground-based support facilities, and on to university labs and international partner sites.

The role of the End-to-End Test Capability (ETC) is to support SSIS development efforts through advanced integration, that is, validation of operations concepts and demonstration of needed levels of interoperability in system designs. To this end, the ETC activities will require the integration of many SSFP test beds.

Phase I ETC activities consisted of the integration of the JSC DMS Test Bed, the OMA Node, and the GN&C Emulator Test Bed to perform a Space Station Freedom reboost scenario, leaving hooks and scars for adding other systems simulations as they become available. The implementation of Phase I was completed and was widely demonstrated. Phase II ETC integration will functionally integrate several additional system test beds and support nodes into the reboost scenario used in Phase I. Implementation of Phase II is in progress. Future ETC integration efforts will emphasize nodes that are remote to JSC and additional scenarios. The DMS Test Bed will become the equivalent of one node of a broader SSIS ETC.

Systems Autonomy Demonstration Project. The Systems Autonomy Demonstration Program (SADP) is funded by the NASA Office of Aeronautics and Space Technology as part of the SATP. This program's goal is to develop, integrate, and demonstrate the technology to enable intelligent autonomous systems for future NASA missions. The program objectives will be accomplished by a Core Technology research program closely coupled with several major demonstrations, the SADP. The SADP activities include a sequence of progressively more complex demonstrations. This sequence includes the intelligent control and operation of single subsystems in 1988, multiple subsystems in 1990, hierarchical multiple subsystems in 1993, and distributed multiple subsystems in 1996.

The 1988 single system demonstration will be a joint effort between ARC and JSC for the autonomous thermal control system operations for Space Station Freedom. The 1990 multiple system demonstration will be a joint effort between ARC, LeRC, MSFC, and JSC for the autonomous control of the thermal and electrical power systems for Space Station Freedom. The 1993 hierarchical demonstration and the 1996 distributed demonstration will evaluate and validate methodologies for expert system control of multiple subsystems through hierarchical and distributed architectural strategies, respectively.

Space Station Control Center Test Bed. The Space Station Control Center (SSCC), an SSFP ground facility at JSC, will be responsible for tactical and execution level planning and integration of manned base and user systems operations, monitoring, control, and configuration management of the Space Station Freedom core systems, storage and retrieval of core systems data, the overall integrity of the manned base, and the safety of the flight crew.

Many SSCC critical design issues can be evaluated in a stand-alone environment; however, the structure of the integrated ETC environment will be utilized to support testing of several levels of SSCC critical design issues and concepts. Initial SSCC Test Bed efforts were focused on connectivity and interoperability to achieve a distributed test bed environment. This connectivity included the SSCC Test Bed,

the DMS Test Bed, and the C&T Test Bed. Data links to the Goddard Space Flight Center (GSFC) and to Stanford University were also established. The ground portion of the OMS, the Operations Management Ground Application (OMGA), is a major piece of the SSCC software, and an OMGA prototype is being implemented.

Other Significant Research Activities

INCO Expert System Project. One part of the SATP consists of specific domain demonstrations. A set of these demonstrations has been planned to facilitate technology transfer to domains other than the Space Station Freedom. One of these demonstrations is the Integrated Communications Officer (INCO) Expert System Support Project (IESP). This demonstration is significant in that it will be the first NASA knowledge-based system to be implemented into a real-time operational environment.

The objective of IESP is to provide assistance to INCOs in the management (i.e., control and monitoring) of two-way communication between the ground and the Space Shuttle in orbit, as well as between the Space Shuttle and its payloads. The INCOs' job is to monitor console displays which present real-time information about the communication links (voice, video, digital data) between the ground and the shuttle, and to send commands to communication devices to keep the links working properly. The INCO expert system attempts to emulate the responses of INCOs to communication stream malfunctions and configuration problems.

Transition Flight Control Room. The Transition Flight Control Room (TFCR) at JSC is an engineering test bed for control room hardware and software systems in a near operational environment. A major function of the TFCR is support for Mission Control Center (MCC) Upgrade, the augmentation and replacement of major MCC hardware and software. The TFCR also serves as a generalized control center test bed environment. It provides a demonstration facility for design approaches, allows the validation of user operational requirements, and the transition of new technologies into flight control rooms. The TFCR allows flight controllers to evaluate proposed upgrades under near operational conditions (e.g., using real telemetry data).

Issues Related to Design and Research Facilities. The major issues noted with regard to design and research facilities deal with cooperative problem solving, technology transfer, and communication between these facilities. While the authors were visiting lab and test beds to collect information, personnel at many of these facilities were interested in any information they could collect on activities at other facilities. A formal means of gathering this information could not be identified by these personnel. While communication between some facilities was taking place or was planned, for technology transfer or coordination of schedules (e.g., for SADP demonstrations), poor communication between these facilities could lead to a lack of technical standards, duplication of effort, poorly defined interfaces, scheduling problems, and increased cost. Formal mechanisms by which effective communication and cooperative problem solving can take place, and information can be disseminated, must be defined.

OPERATIONAL AND SUPPORT CAPABILITIES

The operational and support capabilities discussed are the Software Support Environment (SSE) and the Multi-System Integration Facility (MSIF). The SSE is an environment and a set of policies, procedures, and tools that provide an overall framework for managing software development within the SSFP. The components of the SSE are the SSE Development Facility (SSEDF) and the Software Production Facilities (SPFs).

The SSEDF is a single facility located at JSC at which the SSFP software development policies and tools are baselined and made available to the SPFs. In addition, the SSEDF will serve as a central repository of common models of the Space Station Freedom systems to be used for test and integration.

The SPFs are distributed facilities located at NASA centers and contractor sites for SSFP software development using the policies and tools provided by the SSEDF. Special SPFs, called Software Integration Facilities (SIFs), will be used for intra-system integration.

The MSIF will be the final point for integration and test of SSFP flight software before it is certified for flight readiness. The goal of the MSIF is to test and certify SSFP systems that are coupled, through the DMS, with other systems. The MSIF will utilize both system models and actual systems to perform this testing.

Issues Related to the Operational and Support Capabilities. The SSE provides critical capabilities relative to evolution paths for advanced automation development and testing. The SSE will need to accommodate the development and testing of advanced automation applications by providing appropriate tools and procedures, including appropriate life cycle models and verification and validation procedures. Intra-system integration issues relative to advanced automation will need to be addressed within the SIFs. The SSE will need to evolve as advanced automation technologies mature and become available for use in space systems. The role of the MSIF in promoting the use of advanced automation in space will probably be smaller than the role of the SSE. However, the use of advanced automation in distributed systems will certainly present unique system integration problems.

SUMMARY OF EVOLUTION ISSUES AND STRATEGIES

As advanced automation technologies mature and standards are defined, reliable and cost-effective systems can be built using these new technologies if the laboratories, test beds, SSE, and MSIF are prepared to support their application. Currently, the most mature of the new technologies is Knowledge-Based Systems (KBSs). No longer are KBSs considered magic; they are just a new kind of software. The development activities for KBSs correspond in an approximate way to those performed in a conventional software development project, although they are performed in a somewhat different fashion. To support these differences, existing facilities need not be replaced; they need merely to evolve to support these new technologies.

Two development environments must exist within the SSFP to promote advanced automation technologies to the required level of technology readiness: environments for state-of-the-art technology development and state-of-the-practice application development. For the SSFP, technology development is nurtured in laboratories and test beds; application development is supported in the SSE. Evolution of Space Station Freedom is linked both to the laboratories and test beds as well as to the SSE software factory. These existing facilities should evolve technologically in parallel with the evolution of the Space Station Freedom.

The following evolution issues relating to the existing facilities have been identified and discussed in this review of SSFP capabilities for the promotion of advanced automation:

- A mechanism is needed that facilitates two way communication between facilities to allow co-operative problem solving, interoperability, integration, and coordination of existing and planned SSFP laboratories and test beds.
- Criteria and procedures must be developed for determining the technology readiness of tools before they can emerge from the laboratories to the test beds and subsequently to the SSE.
- A mechanism is required to evaluate commercially available tools and test bed developed prototypes for technology readiness. Doorway tests to gain entrance either to the SSE or the MSIF must be defined.
- SSE support for a spiral/iterative development life cycle model is required for KBS applications where sufficient knowledge is not available for the complete specification of requirements. Specifications are required early in the life cycle of the traditional waterfall life cycle model.

- Development of Verification and Validation (V&V) methods are required for the characteristics of KBSs that make these systems different from conventional technologies. The problems of validating the expert and proving the correctness of the transformation of expert knowledge to code (i.e., rules and facts) must be addressed.

A management approach is required to respond to many of these evolution issues and to manage facility evolution to accommodate emerging technologies. An integration team with a small number of members from the new technology area of KBSs is the proposed management model. As new advanced automation technologies are identified as candidates for the SSFP, additional integration teams would be established. An integration team would have the following responsibilities: establish methods for disseminating the technology knowledge base of that technology; review and evaluate proposals within the technology area; establish hardware and software standards to maximize interoperability and resource sharing among laboratories and test beds; coordinate proposals to facilities, systems and elements; perform peer review of projects involving the technology; and provide feedback to management on these projects.

FUTURE EVOLUTION PATH ACTIVITIES

The information, related analyses, and contacts established in the preparation of this report on SSFP capabilities for the promotion of advanced automation provide a sound foundation for further efforts in defining a comprehensive SSFP advanced automation evolution plan. The objectives of the next phase of this task are viewed to be the definition of evolutionary goals for SSFP test beds and facilities, and the definition of evolutionary paths and plans for reaching these goals. In addition, collective evolutionary goals will be defined for these facilities as a set.

Before the definition of evolution plans, the endpoints (i.e., goals) must be clearly defined. Among the issues to be addressed in determining these goals are distinguishing criteria, roles, and functions; interoperability, commonality, and integration; and the development, demonstration, and delivery environments. Once the destination has been well defined, the fundamental tools needed to get there must be identified. A general road map for SSFP decision making is needed in addition to test bed-specific evolution plans. Information from the Transition Definition Program evolution studies and advanced development tasks, as well as system, test bed, and contractor management will be used to tie these facility evolution plans closely with plans for the evolution of Space Station Freedom's systems. Another necessary integrating activity is involvement in the SSE and MSIF requirements definition process and the development of evolutionary plans for the SSE and MSIF relative to advanced automation.

TABLE OF CONTENTS

SECTION	PAGE
1 Introduction	1
1.1 Background	1
1.1.1 Task Purpose	1
1.1.2 Task Scope	2
1.2 Document Purpose and Scope	2
1.3 Approach	3
1.4 Document Organization	5
2 Design and Research Capabilities	7
2.1 Design and Research Test Beds	7
2.1.1 Data Management System Test Beds	8
2.1.1.1 JSC DMS Test Bed	8
2.1.1.2 IBM DMS Test Bed	11
2.1.2 JSC Operations Management System Prototypes	13
2.1.2.1 Integrated Status Assessment Prototype	14
2.1.2.2 Procedures Interpreter Prototype	15
2.1.3 Thermal Control System Test Beds	15
2.1.3.1 JSC Thermal Test Bed	16
2.1.3.2 ARC Thermal Test Bed	17
2.1.4 Electrical Power System Test Beds	18
2.1.4.1 LeRC Automated EPS Test Bed	19
2.1.4.2 MSFC Power Management and Distribution System Test Bed	23
2.1.4.3 JSC Generic Electrical Power Distribution and Control Test Bed	27
2.1.5 JSC Guidance, Navigation, and Control Emulator Test Bed	27
2.1.6 JSC Communications and Tracking System Test Bed	29
2.1.7 ARC Advanced Architecture Test Bed	30
2.1.8 Other DMS Nodes	33
2.1.8.1 JSC Human Computer Interaction Node	33
2.1.8.2 JSC MPAC Displays and Control Node	33
2.2 Major Test Bed Integration Activities	35
2.2.1 End-to-End Test Capability	35
2.2.2 Systems Autonomy Demonstration Project	39
2.2.2.1 1988 Single System Demonstration	39
2.2.2.2 1990 Multi-System Demonstration	39
2.2.2.3 1993 Hierarchical System Demonstration	40
2.2.2.4 1996 Distributed System Demonstration	40
2.2.2.5 Issues	41
2.2.3 Space Station Control Center Test Bed	41
2.3 Other Significant Research Activities	43
2.3.1 INCO Expert System Project	43
2.3.2 Transition Flight Control Room	45
3 Operational and Support Capabilities	47
3.1 Software Support Environment	47
3.1.1 Software Support Environment Development Facility	49

TABLE OF CONTENTS (continued)

SECTION	PAGE
3.1.1.1 General Description	49
3.1.1.2 Advanced Automation Capabilities	49
3.1.2 Software Production Facilities	50
3.1.2.1 General Description	50
3.1.2.2 Advanced Automation Capabilities	51
3.1.3 Verification and Validation Capabilities	51
3.1.3.1 Verification and Validation Process	51
3.1.3.2 Testing Approach	53
3.1.3.3 Testing Architecture	55
3.1.3.4 Advanced Automation Capabilities	55
3.2 Multi-System Integration Facility	55
3.2.1 General Description	56
3.2.2 Advanced Automation Capabilities	57
4 Evolution Paths for Advanced Automation	57
4.1 Technology Evolution	60
4.2 Facility Evolution Strategies	60
4.2.1 The Incorporation of New Technologies	61
4.2.2 The Need for Two Development Environments	61
4.2.2.1 Laboratories and Test Beds: State-of-the-Art Technology Development	62
4.2.2.2 SSE: State-of-the-Practice Development	62
4.2.3 Implications for the Laboratories and Test Beds	62
4.2.3.1 Research Laboratories	62
4.2.3.2 Design Test Beds	63
4.2.4 Implications for the SSE	64
4.2.4.1 An SSEDf Expert System Shell	65
4.2.4.2 Life Cycle Models	65
4.2.4.3 KBS Verification and Validation	66
4.2.5 An Organizational Model for Technology Development Evolution	67
4.2.6 An Organizational Model for Application Development Evolution	67
4.2.6.1 Development Community Readiness or Awareness	68
4.2.6.2 Advanced Automation Technical Review Team	69
4.2.6.3 Dollar Signature Limits	69
4.2.6.4 Procurement Cycles	69
4.3 Application Evolution	69
4.3.1 Application Development	69
4.3.1.1 Pre-SSE Application Development	70
4.3.1.2 Post-SSE Application Development	70
4.3.2 Intra-System Integration	70
4.3.3 Multi-System Integration	70
4.3.4 Ground Testing	70
4.3.5 Onboard Operations	72
4.4 Summary of Evolution Issues and Strategies	73
4.5 Future Evolution Path Activities	73

TABLE OF CONTENTS (Continued)

SECTION	PAGE
Appendix A: Advanced Automation	75
A.1 Applications of Advanced Automation	75
A.2 Advanced Automation Technologies	75
A.2.1 Expert Systems and Knowledge Based Systems	75
A.2.1.1 Logic-Based Systems	78
A.2.1.2 Semantic Net Systems	79
A.2.1.3 Rule-Based/Production Systems	79
A.2.1.4 Blackboard Systems	79
A.2.1.5 Model-Based Systems	79
A.2.1.6 Frame-Based Systems	79
A.2.1.7 Hybrid Systems	80
A.2.2 Emerging Advanced Automation Technologies	80
A.2.2.1 Artificial Neural Systems	80
A.2.2.2 Fuzzy Logic-Based Systems	82
A.2.3 Advanced Automation Interfaces	83
A.2.3.1 Natural Language Understanding	83
A.2.3.2 Continuous Speech Recognition	83
A.2.3.3 Speech Synthesis	84
A.2.3.4 Vision and Image Processing	84
A.3 Candidate Applications for Space Station Freedom	84
A.4 Advanced Automation Technology Readiness	85
A.4.1 Technology Assessment Curves	85
A.4.2 Candidate Application Technology Readiness	88
Appendix B: Software Engineering for Advanced Automation	89
B.1 Software Engineering Activities	89
B.1.1 Concept Formulation	89
B.1.2 Requirements Definition Activities	89
B.1.3 Design Activities	90
B.1.3.1 Architecture Selection	90
B.1.3.2 Knowledge Representation	90
B.1.3.3 Tool Selection	91
B.1.4 Development Activities	92
B.1.4.1 Code	92
B.1.4.2 Unit Test on Development System	92
B.1.4.3 Port to Delivery System	92
B.1.5 Unit Test on Delivery System	93
B.1.6 Independent Verification/Acceptance Activities	93
B.1.7 Operations Activities	93
B.1.7.1 Configuration Management	93
B.1.7.2 Maintenance	93
B.2 Software Life Cycle Models	94
B.2.1 Waterfall Model	94
B.2.2 Iterative Development Model	95
B.2.3 Spiral Model	96
B.2.4 Transform Model	98

TABLE OF CONTENTS (Concluded)

SECTION	PAGE
Appendix C: Key Contacts	101
List of References	105
Glossary	111
Distribution List	115

LIST OF FIGURES

FIGURE	PAGE
2-1 DMS Test Bed Network	9
2-2 DMS Test Bed Network Protocol Model	10
2-3 Multipurpose Interface Device 1553 Simulation Sub-Unit	12
2-4 Integrated Status Assessment Prototype Schematic and Interfaces	14
2-5 Procedures Interpreter Prototype Schematic and Interfaces	16
2-6 Simplified Schematic of the JSC Thermal Test Bed Hardware	17
2-7 TEXSYS Architecture	18
2-8 LeRC Automated Space Station Freedom EPS Test Bed	19
2-9 Power System Automation (Initial Space Station Freedom)	21
2-10 LeRC EPS APEX Automation Concept	22
2-11 MSFC SSM/PMAD Test Bed	24
2-12 SSM/PMAD Automation Architecture	25
2-13 Generic Electrical Power Distribution and Control Test Bed	27
2-14 GN&C Test Bed	28
2-15 C&T Control and Monitoring Test Bed	30
2-16 Advanced Architecture Test Bed	31
2-17 Human Computer Interaction (HCI) Test Bed	34
2-18 MPAC Display and Control Node	32
2-19 ETC Phase I Integration	37
2-20 ETC Phase II Integration	38
2-21 1990 SADP Demonstration: Cooperating Systems	40
2-22 SSCC Test Bed Architecture	42
3-1 Operational and Support Facilities	48
3-2 SSE Reference Architecture	49
3-3 Integration, Test and Verification	52
3-4 TAVERNS Concept	54
3-5 DMS Kits Block Diagram	55
4-1 Advanced Automation Technology Evolution	58
4-2 Possible Application Evolution Paths	70
A-1 General Architecture of Knowledge-Based Systems	77
A-2 Example Neural Network Model	81
A-3 Fuzzy Set Membership: Small, Medium, and Large Integers	82
A-4 Computing Technology Assessment Curve	86
A-5 KBS Technology Assessment Curve	86
A-6 Planning Technology Assessment Curve	87
A-7 Speech Technology Assessment Curve	87
B-1 Waterfall Model of the Software Life Cycle	95
B-2 Spiral Model of Software Life Cycle	97
B-3 Transform Life Cycle Model	98

LIST OF TABLES

TABLE		PAGE
A-1	Categories of Advanced Automation Applications	75
A-2	Advanced Automation Architectures, Techniques, and Interfaces	78
A-3	Autonomous System Manager Level 6 Readiness	88
A-4	Fault Predictor/Analyzer/Manager Level 6 Readiness	88

SECTION 1

INTRODUCTION

1.1 BACKGROUND

The use of advanced automation and robotics within the Space Station Freedom Program is motivated by potential benefits, such as reduced life cycle costs and technology transfer to industry. As advanced automation and robotics technologies mature, they will be utilized within systems onboard Space Station Freedom and the space platforms, as well as within control centers on the ground.

Systems which utilize advanced automation may differ from standard flight systems developed using conventional, procedural, high-level programming languages and associated environments. Advanced software may be developed within non-standard hardware environments (e.g., Lisp machines, parallel processing systems, or neurocomputers) with non-procedural development tools (e.g., rule-based expert system shells, logic-based languages such as Prolog, or artificial neural systems). Advanced, automated applications utilizing these technologies will need to be tested, validated, and verified before being flight-certified. These technologies will need to be elevated to a high level of readiness before they can be safely used in space.

The National Aeronautics and Space Administration (NASA) has substantial expertise relative to the development and testing of conventional software systems for in-flight use; however, little experience or knowledge exists with respect to engineering, testing, and certifying systems which utilize advanced automation technologies. For this reason, the NASA Space Station Advanced Development Program, Strategic Plans and Programs Division (Code ST), Office of Space Station (OSS) is funding an effort to define evolutionary requirements for ground-based test beds and development facilities related to the coordinated use of advanced automation for Space Station Freedom. The Intelligent Systems Branch, within the Engineering Directorate at the Johnson Space Center (JSC), is directing this task.

The Intelligent Systems Branch has varied responsibilities to Space Station Freedom Levels I, II, and III. The responsibilities center on advising the Space Station Freedom Program (SSFP) on approaches to the use of A&R technology within the SSFP. The Intelligent Systems Branch acts as both an advocate and an evaluator of A&R activities within the SSFP. The MITRE Corporation has been supporting the Intelligent Systems Branch with these activities for several years.

1.1.1 Task Purpose

The purpose of this task is to define evolutionary requirements for SSFP test beds and software engineering, test, and integration facilities, relative to engineering, demonstration, test, integration and subsequent flight certification of systems utilizing advanced automation technologies for safe use onboard Space Station Freedom. The resulting requirements and facilities are intended to promote the science of advanced automation through certain phases of technology readiness. NASA employs the following 8 levels of technology readiness:

- | | |
|---------|---|
| Level 1 | Basic principles observed and reported |
| Level 2 | Conceptual design formulated |
| Level 3 | Conceptual design tested analytically or experimentally |
| Level 4 | Critical function or characteristic demonstration |
| Level 5 | Component or breadboard tested in relevant environment |
| Level 6 | Prototype or engineering model tested in relevant environment |
| Level 7 | Engineering model tested in space |
| Level 8 | Full operational capability (incorporated in production design) |

Automation technology, which is at level 4 or 5, needs to be moved on to level 6 before it is ready for in-space testing. The automation technologies which are at level 4 or 5 must be identified so they can be considered as candidates for testing. The facilities defined or affected by this task will enable automation technologies to be brought to a level 6 state of readiness.

The original thrust of this task was toward the definition of requirements for one or more separate Advanced Automation Test Beds. As the study progressed, the authors came to believe that augmentation of existing test beds and support facilities was a better solution to SSFP advanced automation needs for two reasons. First, advanced technologies will need to interface with or be integrated with conventional technologies. Second, separate Advanced Automation Test Beds might duplicate or compete unnecessarily with existing laboratories and test beds. This change of focus is compatible with increased interest from SSFP managers in the evolution of existing facilities in support of advanced automation technologies. Therefore, the thrust of the task changed from the definition of requirements for one or more separate Advanced Automation Test Beds to the definition of evolution path requirements for existing capabilities.

1.1.2 Task Scope

The Intelligent Systems Branch originally asked The MITRE Corporation to generate functional and facility requirements, as well as costs and schedules, for advanced automation test beds over a three to six-year period. The task was subsequently redirected toward evolution paths for advanced automation. For the purposes of this task, advanced automation is defined as the use of concepts and methods of high-level symbolic inference by a computer and the symbolic representation of the knowledge to be used in making inferences so as to make a machine behave in ways that humans recognize as intelligent behavior. The technical scope of this study is restricted to advanced automation only and, therefore, excludes robotics.

The goal of advanced automation is to assist and/or automate the human control or decision making role in complex processes where previous attempts have proven impossible, unrealistic, or non-cost-effective. The key concept is the mimicry of the human inferencing, not the techniques used to implement this behavior. Therefore, advanced automation includes the use of expert systems, knowledge-based systems, and other techniques from the field of artificial intelligence (AI), but does not exclude the use of conventional hardware and software.

1.2 DOCUMENT PURPOSE AND SCOPE

Before functional requirements for the evolution of SSFP facilities supporting software engineering of advanced automation applications can be defined, a solid understanding of advanced automation and SSFP capabilities and plans for advanced automation is necessary. The purpose of this document is to review, for SSFP managers, SSFP automation capabilities and plans, including facilities, applications, and existing prototypes. This document will serve as a basis for subsequent requirements definition efforts.

Expert systems provide one of the first commercially viable technologies to emerge from the AI labs of the nation's universities and research centers. This technology is being used within industry and the government. Expert systems, especially rule-based systems, are the most likely candidates for early use of advanced automation applications onboard Space Station Freedom. For these reasons, this report concentrates on expert systems. Expert systems attempt to emulate human expertise. Knowledge-based systems (KBSs) are similar to expert systems, except that they do not specifically model human expertise. Nevertheless, the two names are used synonymously in this document. Knowledge-based systems and other advanced automation technologies are introduced and discussed in Appendix A.

1.3 APPROACH

This work was divided into the following activities:

- A survey of the advanced automation literature, including
 - Software engineering and software life cycles for expert systems
 - Verification and validation (V&V) of expert systems
- A survey of SSFP documentation, including
 - Facility definitions
 - V&V plans
 - Advanced automation plans and reports
- Visits to advanced automation test beds and laboratories, including
 - NASA facilities
 - Academic research facilities

Several studies have preceded this task and provide a firm foundation for the definition of advanced automation software engineering facilities for the SSFP. Information resulting from these and other studies has been used as a basis for this task and to avoid duplication of effort. The following studies are especially important:

- **Ongoing reports of the NASA Advanced Technology Advisory Committee (ATAC).** In response to the interest of Congress in promoting advanced automation and robotics, ATAC was tasked to consider the needs and limitations of the SSFP and the findings of other studies, and to provide recommendations to NASA. ATAC reports to Congress (TM-87566, March 1985).
- **Systems Autonomy Technology Program.** The OAST has implemented through the NASA Ames Research Center (ARC) a Systems Autonomy Technology Program (SATP) that sponsors and pursues the required research, developments, and technology demonstrations for integration of intelligent autonomous systems into space systems. These efforts include both core technology research and the aggregation and integration of these core technologies into meaningful technology demonstration projects where prototype systems will be tested in the context of realistic application scenarios to assure technology relevancy and maturity for space mission applications (ARC, November 1987^a).
- **Space Station Technology Development Mission Requirements Definition for Advanced Automation Study.** A report by the Boeing Aerospace Company defines A&R development missions for the SSFP. The study ranked these missions according to criteria that reflect (1) available technology, (2) the economics of using Space Station Freedom, and (3) the usefulness of the A&R applications supported. This study also provided pointers to space station facilities and capabilities, and Initial Operations Capability (IOC) hooks and scars needed for the high-priority missions (Boeing, 1987).
- **Space Station Advanced Automation Study Final Report.** The purpose of this study was to perform a rapid analysis of the current and future potential of knowledge-based systems (often called expert systems) on Space Station Freedom. This study made the following recommendations pertinent to our efforts: (1) a short list of baseline space station candidate applications feasible with current technology, including cost and performance estimates, and (2) an

analysis of knowledge-based system evolution on Space Station Freedom, including descriptions of recommended hooks and scars to be provided at baseline (Friedland et al., May 1988).

The MITRE Corporation's support of Space Station Freedom development has included support for the development of the Data Management System (DMS), Operations Management System (OMS), Space Station Control Center (SSCC), Space Station Information System (SSIS), Software Support Environment (SSE), Multi-System Integration Facility (MSIF), Space Station Training Facility (SSTF), and others. In addition, The MITRE Corporation has provided support to NASA on National Space Transportation System (NSTS) efforts for many years. To extend this broad knowledge base in support of this task, both NASA and non-NASA advanced automation test beds and laboratories were visited. Time, man-power, and scheduling constraints limited this tour to the following sites:

- Design and Research Test Beds
 - Data Management System Test Beds
 - Johnson Space Center Data Management System Test Bed
 - International Business Machines Data Management System Test Bed
 - Operations Management System Prototypes
 - Thermal Control System Test Beds
 - Ames Research Center Thermal Control System Test Bed
 - Electrical Power System Test Beds
 - Lewis Research Center Electrical Power System Test Bed
 - Marshall Space Flight Center Space Station Module/Power Management and Distribution Test Bed
 - Ames Research Center Advanced Processor Test Bed
- Major Test Bed Integration Activities
 - Systems Autonomy Demonstration Program Facilities
 - Space Station Information System End-to-end Test Capability
- Research Laboratories
 - Ames Research Center
 - Lewis Research Center
 - Marshall Space Flight Center
 - Autonomously Managed Power Systems Laboratory Facilities
 - Johnson Space Flight Center
 - Integrated Communications Officer Expert System Project
 - Transition Flight Control Room
 - Carnegie Mellon University
 - University of Texas at Austin

Where constraints did not allow personal visits to observe the activities, information on test beds, laboratories, and prototyping efforts was derived from available documentation, phone conversations with responsible personnel, and interviews with MITRE personnel familiar with these efforts.

1.4 DOCUMENT ORGANIZATION

The remainder of the document reviews SSFP facilities, capabilities, and plans relative to the development and use of advanced automation onboard Space Station Freedom. Section 2 reviews SSFP research and design capabilities, existing SSFP laboratories, and test beds for onboard and ground systems. Currently, most of the thought and activities related to advanced automation take place within these facilities. Section 3 reviews operational and support capabilities of the SSFP with respect to formal software development, test, and integration, i.e., the SSE and the MSIF. These are the facilities that must evolve to support the application of mature technologies. Section 4 discusses evolution paths for promoting advanced automation applications toward flight readiness onboard Space Station Freedom. Supporting information is included in the appendices. Appendix A introduces and defines advanced automation technologies and applications, defines candidate applications for Space Station Freedom, and introduces technology readiness levels. Appendix B discusses software engineering and software life cycles for advanced automation applications. Appendix C lists the key contacts interviewed during this study.

SECTION 2

DESIGN AND RESEARCH CAPABILITIES

This section addresses the advanced automation design and research capabilities of the Space Station Freedom Program (SSFP). Section 2.1 discusses the design and research capabilities of existing SSFP laboratories and test beds for onboard and ground systems. Section 2.2 discusses major test bed integration activities. These include the End-to-End Test Capability, the Systems Autonomy Demonstration Project, and the Space Station Control Center Test Bed. Because there are references to these integration activities in Section 2.1, they are introduced briefly below. Section 2.3 presents information on other significant advanced automation research efforts.

End-to-End Test Capability. The Space Station Information System (SSIS) comprises the information processing and communications capabilities that will be involved in handling operational and scientific data generated or used within the SSFP. The SSIS will provide end-to-end connection of, and a variety of information services to, a diverse and geographically distributed user community. This end-to-end connectivity will reach from the flight systems and payloads, to the ground-based support facilities, and on to university labs and international partner sites. The role of the End-to-End Test Capability (ETC) in the SSFP is to support SSIS development efforts through advanced integration; that is, validation of operations concepts and demonstration of needed levels of interoperability in system designs. The ETC will integrate many SSFP test beds.

Systems Autonomy Demonstration Project Summary. The Systems Autonomy Demonstration Project (SADP) is funded by the NASA Office of Aeronautics and Space Technology (OAST) as part of the Systems Autonomy Technology Program (SATP). This program's goal is to develop, integrate, and demonstrate the technology to enable intelligent autonomous systems for future NASA missions. The program objectives will be accomplished by a Core Technology research program closely coupled with several major demonstrations, the SADP. The SADP activities include a sequence of progressively more complex demonstrations. This sequence includes the intelligent control and operation of single subsystems in 1988, multiple subsystems in 1990, hierarchical multiple subsystems in 1993, and distributed multiple subsystems in 1996.

While the SADP project is not directly funded by the SSFP, these demonstration activities are closely related to the design of the Space Station Freedom onboard systems and have been and will continue to be intimately involved with the SSFP test beds and domain experts.

Space Station Control Center Test Bed. The Space Station Control Center (SSCC) will be the ground operations center for the SSFP and will coordinate and manage the operations of the Space Station Freedom flight systems as well as many ground systems. SSCC Test Bed activities will integrate many of the SSFP test beds in order to test possible SSCC critical design issues and operations concepts.

2.1 DESIGN AND RESEARCH TEST BEDS

The SSFP has funded, mainly through advanced development funds, the development of test beds for many of the Space Station Freedom ground facilities and flight systems. These test beds provide facilities that include the basic capabilities needed to evaluate critical system design issues and operations concepts. In addition, they provide environments in which to test needed levels of interoperability with other systems, and also to provide for technical evaluations of commercial products. The following paragraphs discuss the current capabilities and future plans of these test beds.

2.1.1 Data Management System Test Beds

The Data Management System (DMS) will provide hardware resources and software services on the manned base that support the data processing and communications requirements of Space Station Freedom systems and flight elements. It will function as an integrating entity providing a common operating environment and human-machine interface for the operation and control of the orbiting Space Station Freedom systems and payloads by both the crew and the ground operators.

DMS will provide a family of compatible computers ranging from a single-board computer suitable for use as an embedded controller to a general purpose processor suitable for hosting system application software. Each processor will have a compatible set or subset of the DMS operating system. DMS communication and networking components will provide the *glue* that holds the processing, workstation, and sensor/effector environments together. DMS will provide several high rate local area networks as well as lower rate networks and buses to satisfy both the high-bandwidth requirements of payload experiments and the stringent response-time requirements of core subsystems (SSP, 1988^b). There are two DMS test beds: the Johnson Space Center (JSC) DMS Test Bed oriented toward functions and services and the International Business Machines (IBM) DMS Test Bed oriented toward hardware design testing. Throughout this document, unprefaced references to the *DMS Test Bed* refer only to the JSC DMS Test Bed.

2.1.1.1 JSC DMS Test Bed

The JSC DMS Test Bed is managed by the Flight Data Processing Branch of the Avionics Systems Division within the Engineering Directorate at JSC. The test bed was established to evaluate data management, processing, and system control techniques in a distributed system environment in support of SSFP requirements. Development and testing of network operating system software, integrating the network with attached DMS Test Bed nodes, is also a major activity of the test bed (LEMSCO, 1988). Communication networking and services are provided between multiple subsystems (test beds). Currently, the core of the DMS Test Bed consists of an Apollo Computer Inc. token-passing ring network which provides this communication capability. The test bed will transition to Fiber Distributed Data Interface (FDDI) sometime between May 1989 and January 1990. The FDDI standard specifies the protocols for a 100-Mbps fiber ring network.

Physical Test Bed. Figure 2-1 is a schematic of the DMS Test Bed network, including interfaces to DMS Test Bed nodes. These nodes are connected by the DMS Test Bed fiber-optic token ring local area network (LAN). Transmission Control Protocol/Internet Protocol (TCP/IP) is used as the network routing software. Apollo Domain Server Processor (DSP80) computers serve as Network Interface Units (NIUs) (and Bus Interface Units--BIUs) providing a gateway to subsystem applications running on end user nodes. Each NIU is connected via an Ethernet link to the end user node computer which it serves. End user nodes currently include the following computers:

- DEC MicroVAX running VMS
- DEC MicroVAX running UNIX
- Sun workstations running UNIX
- Symbolics Lisp workstations

The Digital Equipment Corporation (DEC) MicroVAX is currently the test bed Standard Data Processor (SDP) and is fully supported for communication services; the other end user nodes listed are not fully supported and most applications residing on them are being converted to run on the DEC machines. Support for other computers is currently undefined. The test bed will transition later to Intel 80386 processors to play the role of the SDPs.

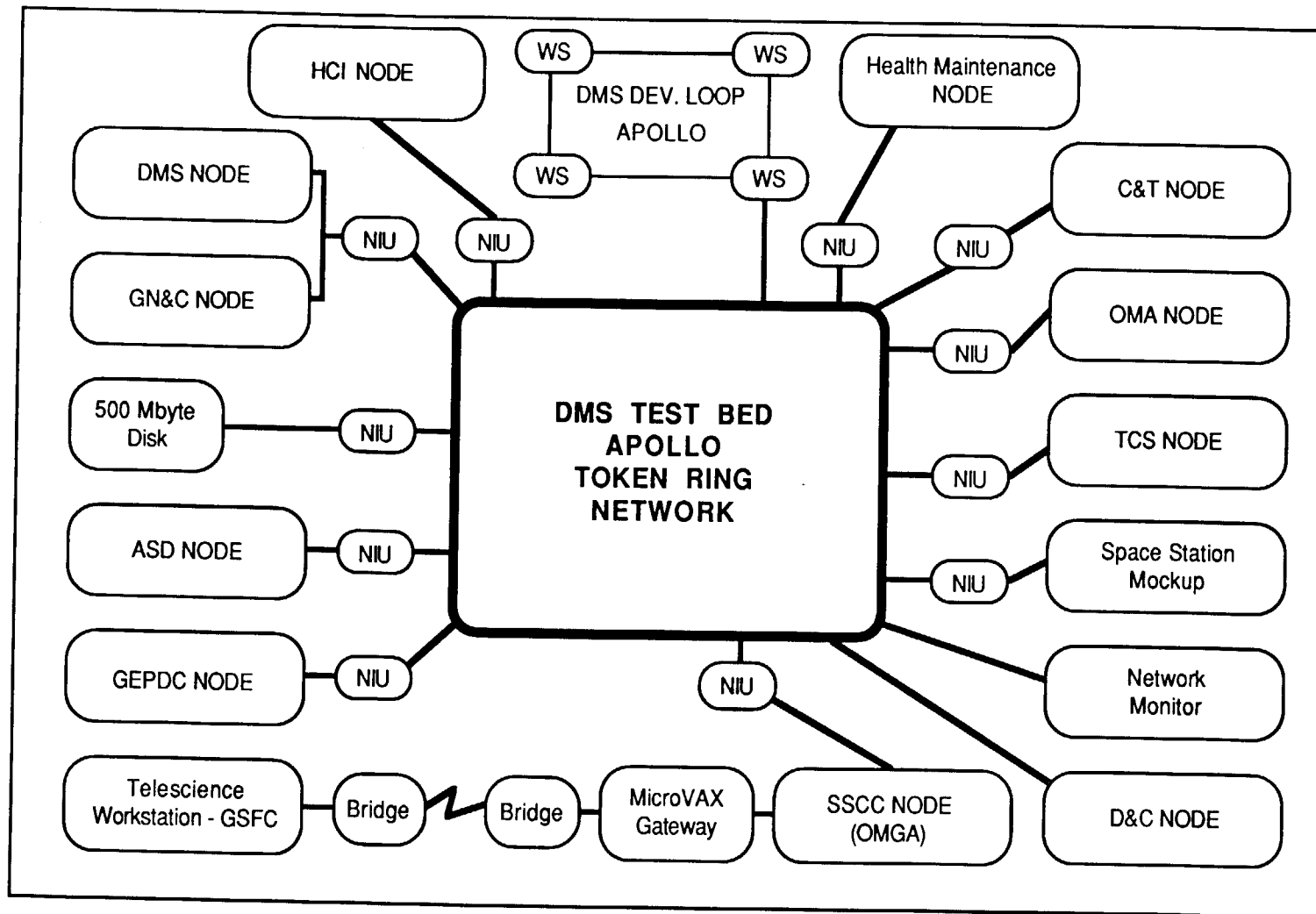


Figure 2-1. DMS Test Bed Network

Also integrated into the network are two Domain Computational Nodes, a DN300 and a DN3000, and a File Server DSP80. The DN300, known as the Loop Control, is used to monitor and control the operation of the network. The DN3000, which supports high-resolution color graphics, is used to develop Crew Work Station network applications in an effort to test and demonstrate the use of network services. Attached to the File Server DSP80 is a 500 megabyte disk which provides access to software and data (LEMSCO, 1988).

Software. The network communications software for the DMS Test Bed is referred to as the Network Operating System (NOS) and is being developed by Lockheed. The NOS provides guaranteed message delivery, directory service, authorization checks, and other services to applications supported by the NOS Library (NOSLIB), a set of library procedures providing access to the NOS. As illustrated in Figure 2-2, the NOS comprises the Network Interface Element (NIE) and the Network Service Element (NSE). The NIE resides in the subsystem computers while the NSE resides in the Apollo NIU (LEMSCO, 1988).

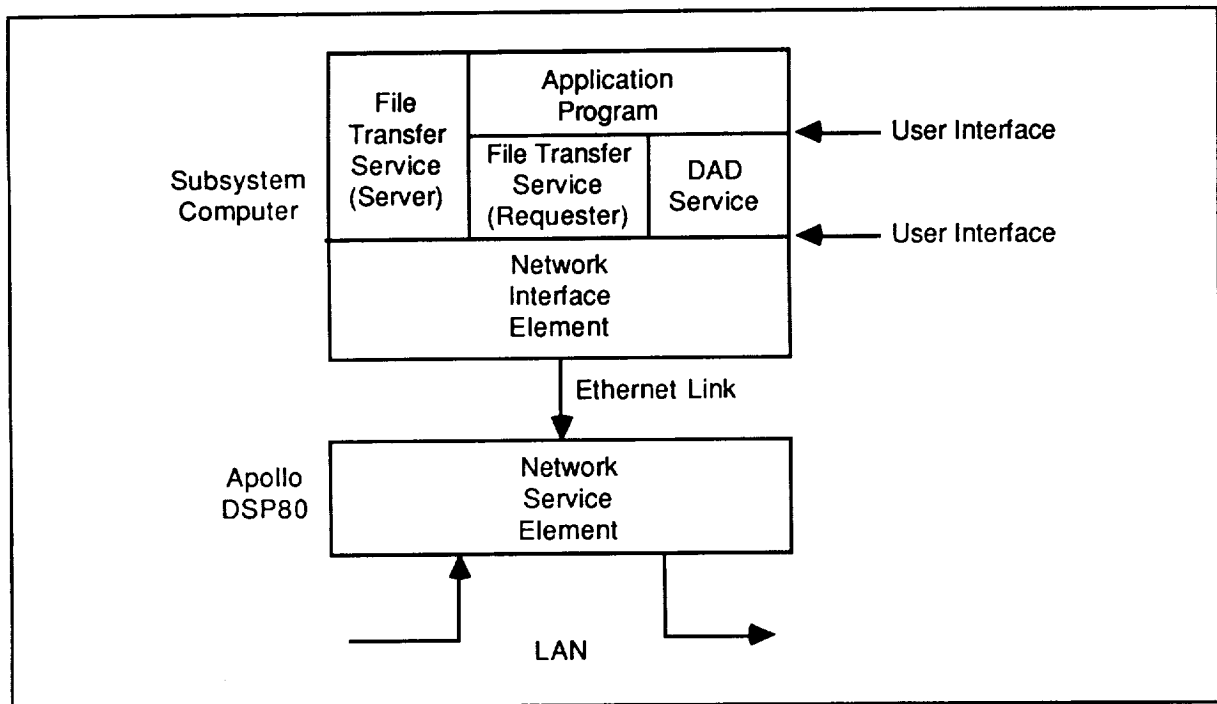


Figure 2-2. DMS Test Bed Network Protocol Model
(Source: LEMSCO, 1988)

Currently available DMS services include Network Communications Services, Data Acquisition and Distribution Services (DADS), Ancillary Data Services (ADS), and File Transfer Services (FTS). The Network Communications Services are provided by the NOS. The DADS, ADS, and FTS represent layers of software situated above the NOS and make use of the communications services provided by the NOS (LEMSCO, 1988). These software services allow the exchange of certain specialized categories of information between subsystems that support these libraries.

Application software consists of Pascal, Lisp, and Ada code. Older network software, e.g., TCP/IP, is written in Pascal. All new software development activities relative to network communication services use Ada.

The current focus of the DMS Test Bed is a transition to a complete Open Systems Interconnect (OSI) network model using Ada and running on an Intel 80386 processor-based architecture. The 80386 will play the role of the test bed's SDP. The test bed possesses one 80386 processor and is currently developing a gateway from the Apollo network to the 80386. The OSI model will probably be implemented on the Apollo systems first because of the experience base that already exists. The current NOS will then be discarded and replaced with OSI. The OSI network operating system and a Verdex Ada compiler are currently on order.

The test bed has just started to use the X-Windows window interface software and is creating an Ada application to run on a DEC MicroVAX with all displays using X-Windows. It is possible that X-Windows will be used as the standard SSFP user interface software.

Integration. The primary users of the DMS Test Bed are the Operations Management System (OMS) prototyping and testing efforts (see Section 2.1.2), the developers of the ETC (see Section 2.2.1), the SADP multi-system demonstration (see Section 2.2.2), and the SSCC Test Bed activities (see Section 2.2.3). In each of these activities the DMS Test Bed serves in an integrating capacity.

Currently, the only link from the DMS Test Bed to other Centers is one to Goddard Space Flight Center (GSFC). The only planned link is the European Space Technology Center (ESTEC) in Holland to connect with their Columbus Module test bed.

2.1.1.2 IBM DMS Test Bed

Contrasted with the functional and service orientation of the JSC DMS Test Bed, which is concerned mainly with providing DMS services and networking capabilities to multiple users, is the IBM DMS Test Bed. This facility is oriented more toward design testing through the development of hardware prototypes relative to actual flight systems concepts proposed by IBM in their Work Package 2 (WP2) proposal. IBM is a subcontractor to McDonnell Douglas on the WP2 contract.

The use of commercial off-the-shelf (COTS) products wherever possible is stressed by IBM Test Bed personnel. IBM is looking into COTS products for the DMS Network Operating System. Test Bed activities include the development of the Multi-Purpose Applications Console (MPAC), Standard Data Processor (SDP), Multipurpose Interface Device (MID), and FDDI prototypes.

Multi-Purpose Applications Console Prototypes. The first MPAC prototype developed by IBM for the National Space Transportation System (NSTS) is based on PC AT technology. The second prototype is a modified version of the NSTS prototype for Space Station Freedom. Both of these prototypes exhibit the use of multiple input devices (e.g., keyboard, touch panel, track ball, voice card, hand controller, etc.). The NSTS version uses separate displays for text, graphics and video. The Space Station Freedom prototype uses one display for simultaneous presentation of text, graphics, and video. Multiple video *windows*, both live and static, can be displayed along with simulated operations scenarios using schematic displays, caution and warning messaging, and strip charts of system parameters. An easily-used human-computer interface, as well as other software, has been developed for these prototypes. COTS hardware (e.g., the PC ATs and the video card) and software (i.e., Turbo Pascal and Turbo C) were used throughout.

The use of COTS products allows the demonstration of these same applications on two other platforms, an AT class machine using a flat panel display, and a prototype workstation based on a PS/2 Model 80 (Intel 80386 architecture), the architecture proposed by IBM for the Station SDP. It should be noted that video was not available for these two prototypes; video cannot be displayed on the flat screen and, to date, no video card is available for the PS/2. The operating system and software (e.g., windowing

software) currently does not meet the IBM proposal which includes the AIX operating system and X-Windows windowing software, but these are targeted for use in the future.

Standard Data Processor Functional Equivalent Unit. IBM is developing a functional prototype of an SDP, based on their SSFP proposal, called a functional equivalent unit (FEU). The SDP FEU will consist of hardware compatible with the PS/2 Model 80 but using different size cards and housing. Each FEU will use the MicroChannel bus for local bus needs and Multibus for global resources. The goal is to achieve complete hardware and software compatibility with the PS/2 Model 80; the machine the authors were shown could run MS/DOS, OS2, and AIX operating systems and related software, including the Model 80 diagnostic applications. Again, the use of COTS products was stressed.

Multipurpose Interface Device 1553 Simulation Sub-Unit. The Multipurpose Interface Device 1553 Simulation Sub-Unit (MID 1553 SSU) is a distributed processing system that will support real-time software development, verification, and training for a candidate processor, or Unit Under Test (UUT). It is assumed that the UUT is based on the Intel 80386 architecture, the architecture of the Station SDP. The MID 1553 SSU looks very much like a DMS Kit (shown in Figure 2-3 and discussed in Section 3.1.3.3) and allows the UUT to operate stand-alone while providing an actual 1553 bus and software to simulate device applications on the other end of the 1553 bus for UUT performance evaluation. The UUT can act as a bus controller, hardware or software based, or a remote terminal. The UUT should be able to perform as if it were a part of its operational environment sending data to and from the 1553 bus unaware that simulations are driving responses and commands to the UUT (Waechter and Walling, 1988).

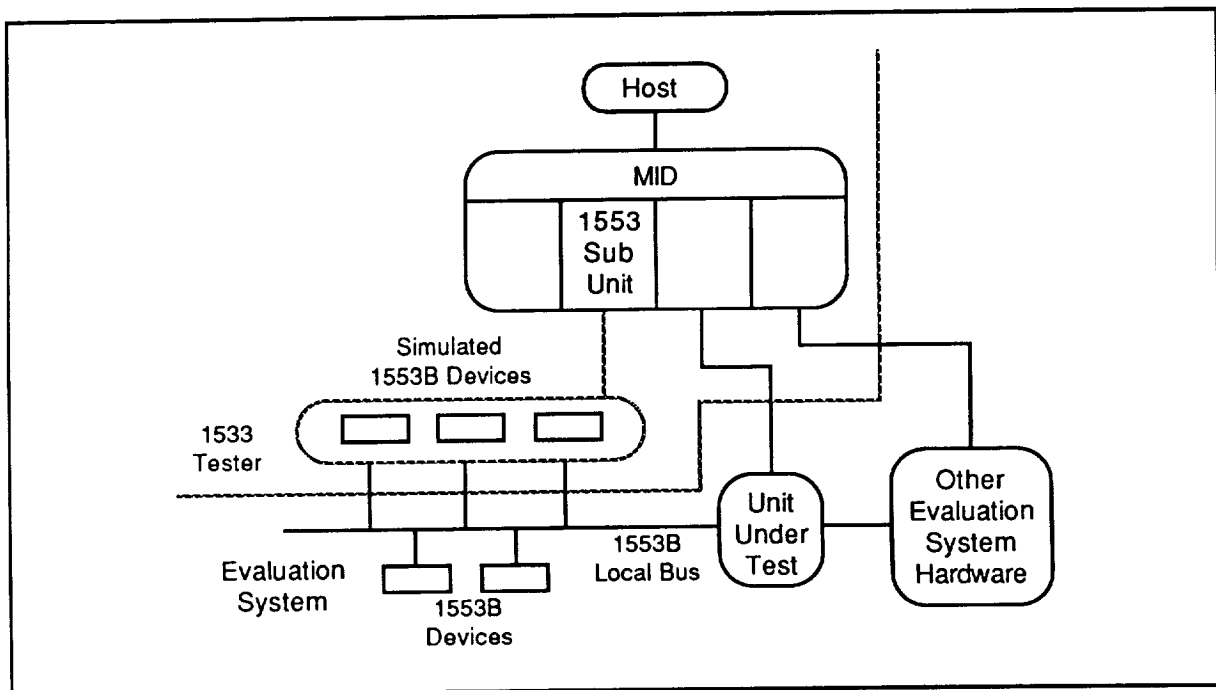


Figure 2-3. Multipurpose Interface Device 1553 Simulation Sub-Unit
(Source: Waechter and Walling, 1988)

FDDI Prototyping. IBM is building prototype networking cards using the FDDI standard. FDDI chips sets have been obtained from the manufacturer, Advanced Micro Devices. These chip sets are expected to be marketed beginning at the end of 1988. At present, a limited first version of an FDDI networking card has been developed by IBM, and a second, more advanced, version is expected to be completed in early 1989. Prototype cards will be delivered to JSC for use in the DMS Test Bed in June of 1989. The present IBM FDDI demo consists of two IBM PC ATs connected by 3 kilometers of fiber optic cable, using FDDI, that repeatedly send graphical data (i.e., pictures) to each other for display.

2.1.2 JSC Operations Management System Prototypes

A major activity which can benefit from increased automation is the command, monitor, and control operations of flight systems; operations management tasks historically performed by the flight crew, ground controllers, and engineering support personnel. Space Station Freedom operations management concepts will be implemented as the Operations Management System (OMS). The OMS will consist of onboard and ground data systems application software, supporting both onboard and ground personnel.

The onboard portion of the OMS is called the Operation Management Application (OMA), a major application of the DMS software. The OMA will be implemented as application software residing on the DMS hardware and will use DMS communications services. The ground portion of the OMS is called the Operations Management Ground Application (OMGA) and will reside within the SSCC.

The OMS is intended to accomplish much of the activity that is presently performed by ground support personnel and thus is expected to become increasingly automated as the Space Station Freedom matures. Advanced automation technologies, as they mature, will be utilized to provide increased autonomy (Eckelkamp and Reilly, 1988).

OMS functions include the following:

- Manage and update the short term plan for Station activities
- Coordinate systems, elements, payloads, and crew operations in execution of the short term plan
- Monitor system, element and payload status
- Manage inter-system, element, and payload testing
- Maintain and log station-wide configuration, activity, and state information
- Detect and manage resource conflicts
- Manage Station-wide caution and warning
- Manage Station-wide fault management and reconfiguration
- Support transaction management
- Provide Station-wide inventory and maintenance management system
- Support onboard training and simulations

Two prototypes, developed by The MITRE Corporation for the Missions Operations Directorate (MOD), are currently being used within the OMS and ETC activities: the Integrated Status Assessment and Procedures Interpreter prototypes. These OMS prototyping activities currently take place on the OMA node of the DMS Test Bed.

2.1.2.1 Integrated Status Assessment Prototype

The Integrated Status Assessment (ISA) prototype performs station-wide failure diagnosis. The OMS functions performed by the ISA prototype include the following:

- Monitor system and payload status
- Maintain and log global configuration, activity, and state information
- Manage global caution and warning
- Perform global fault management and reconfiguration

The ISA prototype expert system's knowledge base consists of facts about Space Station Freedom systems (status and configuration) and rules to perform fault isolation. The prototype was designed as a hybrid system using different methodologies. A model-based representation was used to describe the different systems; rule-based reasoning was used to encode the knowledge on how to perform the failure diagnosis; and qualitative modeling was used to describe the state of the various components (Marsh, 1988). Figure 2-4 shows a schematic and the interfaces of the ISA prototype.

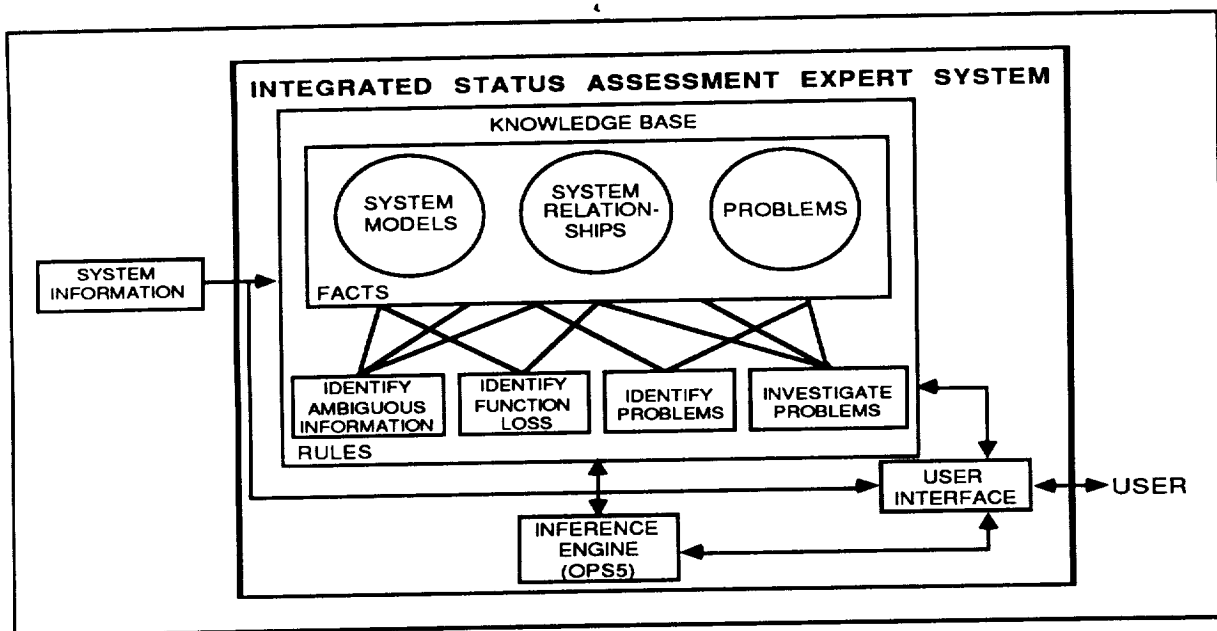


Figure 2-4. Integrated Status Assessment Prototype Schematic and Interfaces
(Source: Marsh, 1988)

It was essential that ISA prototype provide an easy method of building and modifying system models. The user interface is graphical, mouse-driven, and allows the user to build system models, observe system operations, and to control and observe the expert system operations. System models are libraries of components which can be easily created with the mouse and menus, saved to disk, and reloaded into the prototype. A library of graphic symbols can be used to represent components and connections between them.

The ISA prototype is very flexible and can be used to perform diagnosis within many domains. The initial domain chosen for this prototype was the Ku-band portion of the Communications and Tracking

System (C&T). The Ku-band subsystem was chosen because it could be assumed that it would resemble the Shuttle system and therefore a schematic of the system could be derived. In addition, this subsystem intersects with several other systems; for example, the power busses of the EPS, cooling loops of the Thermal Control System (TCS), the Tracking and Data Relay Satellite System (TDRSS), and the DMS network.

The ISA prototype can be operated in several ways. The simplest method allows faults to be manually inserted into a system with subsequent diagnosis of the source of the failure. Simulated or real system data can also be used for more complex operation. Currently, the ISA prototype is being used within the ETC OMS integrated test demonstrations which are discussed in Section 2.2.1.

The ISA prototype was implemented on a Symbolics 3600 series Lisp workstation using Zetalisp and the expert system tool OPS5. At present, ISA is being reimplemented on a microcomputer (IBM-PC/MS-DOS compatible) using the CLIPS expert system shell and the C language. This environment more closely resembles the hardware platform that will ultimately make up the flight systems.

2.1.2.2 Procedures Interpreter Prototype

The Procedures Interpreter prototype demonstrates a possible implementation of one of the OMA functions: execute the short term plan. The short term plan identifies the crew procedures to be executed within certain time and operational constraints. Crew procedures define all structured activities on manned space vehicles. A crew procedure is a defined, tested set of steps that must be taken to operate the vehicle or any item on the vehicle (Kelly, 1988^a; Kelly, 1988^b).

Crew procedures within the NSTS are paper products and are stored in books contained in the Flight Data File. The Flight Data File is referenced by both the flight crew and ground support personnel. Procedures for Space Station Freedom will be stored electronically within the Station Flight Data File. These procedures will be displayed electronically for access by the flight crew. The Procedures Interpreter prototype allows the electronic storage, access, and display, as well as interactive manual and automatic execution, of Space Station Freedom crew procedures.

The major components of the Procedures Interpreter are: the human-computer interface, including the windowing systems and the menu system which drives the user interaction; display definitions, describing what kind of information to display during procedure execution; Station configuration and status information; the procedures; an archiving function to keep a history of the procedures executed and their results; and the Procedure Execution Engine, which accepts the procedures as input and commands the systems to carry out the actions defined in a procedure to be executed. Figure 2-5 shows the logical components of the Procedures Interpreter as well as the external functions which must interact with the Procedures Interpreter (Kelly, 1988^a).

The Procedures Interpreter prototype was implemented on a Symbolics 3600 series Lisp workstation using Zetalisp. It is based on an object-oriented paradigm, using Flavors, with the procedure being the primary object. At present, the Procedures Interpreter is being reimplemented on a DEC MicroVAX using the Operations and Science Instrument Support (OASIS) teleoperations software package. OASIS was developed for NASA by the Operations and Information Systems Division of the Laboratory for Atmospheric and Space Physics at the University of Colorado at Boulder.

2.1.3 Thermal Control System Test Beds

The Thermal Control System (TCS) will maintain the Space Station Freedom equipment and customer payloads within their allowable operational and non-operational temperature ranges. This will be accomplished by passive means unless heat loads and environmental constraints require thermal control

via an active fluid interface. The TCS for the core Space Station Freedom comprises an Active Thermal Control System (ATCS) and a Passive Thermal Control System (PTCS). The ATCS is further subdivided into four active thermal control subsystems. They are the central ATCS, the internal ATCS, the Photovoltaic ATCS, and the Attached Payload Accommodations Equipment (APAE) ATCS (SSP, 1988g).

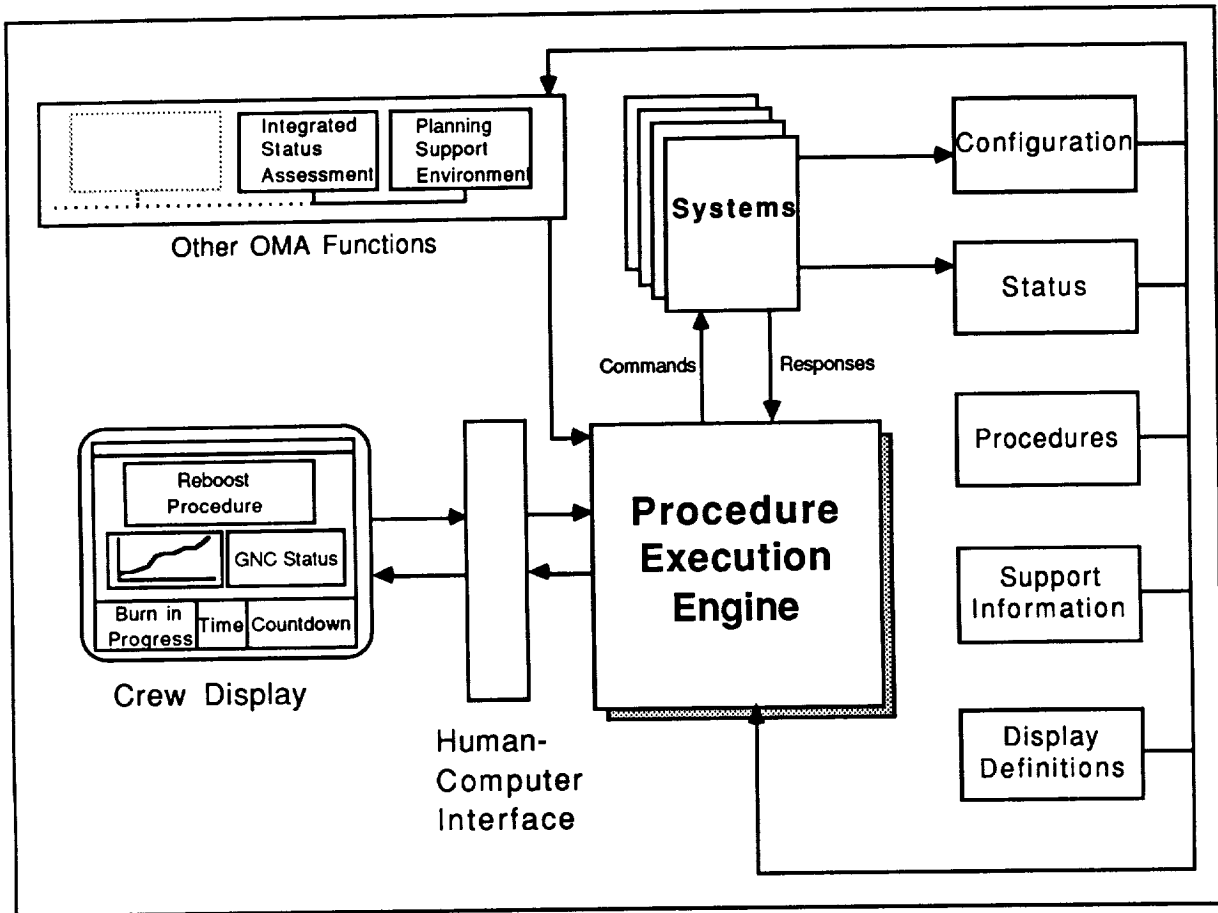


Figure 2-5. Procedures Interpreter Prototype Schematic and Interfaces
(Source: Kelly, 1988^a)

2.1.3.1 JSC Thermal Test Bed

The thermal system for Space Station Freedom is being developed by Crew Systems Division of the Engineering Directorate at JSC. The Thermal Expert System (TEXSYS), developed jointly by thermal engineers at JSC and knowledge engineers at Ames Research Center (ARC), will be used to monitor, control, and diagnose problems on the Thermal Test Bed.

Physical Test Bed. The Thermal Test Bed is a complete thermal system composed of test articles (i.e., pumps, radiators, evaporators, condensers, central thermal buss). Several sets of thermal test articles are being tested and compared. A simplified schematic of an example configuration of the JSC Thermal Test Bed hardware is shown in Figure 2-6.

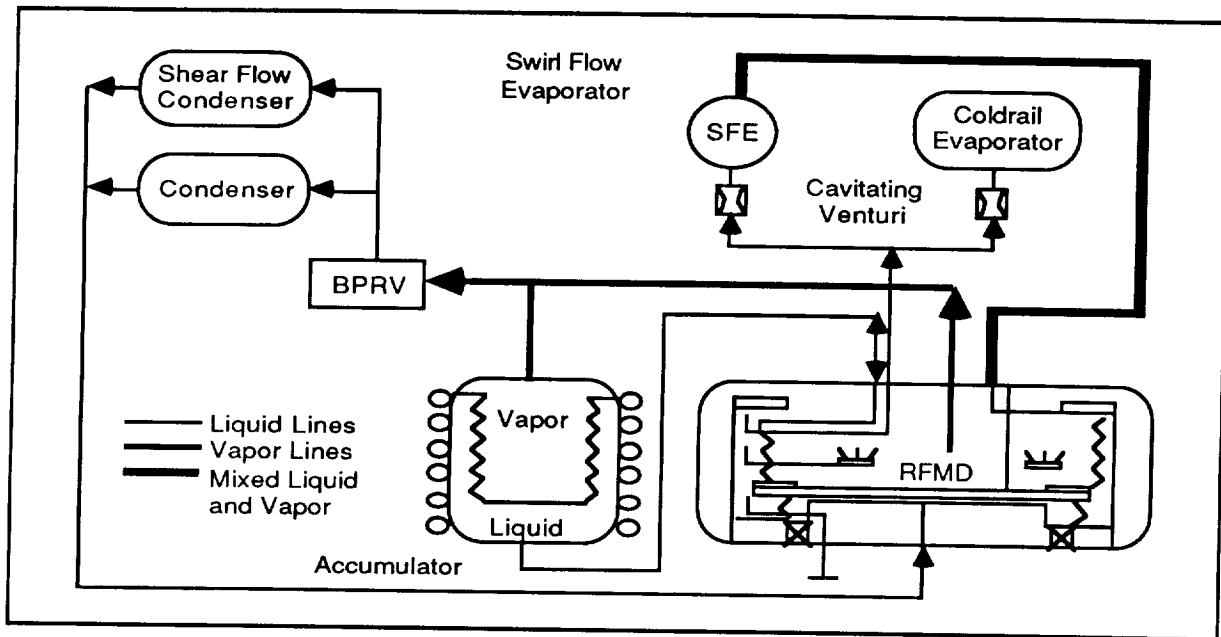


Figure 2-6. Simplified Schematic of the JSC Thermal Test Bed Hardware.
(Source: Wong et al., 1988)

2.1.3.2 ARC Thermal Test Bed

The physical thermal test bed at ARC is a small brassboard of one configuration of the JSC Thermal Test Bed. Computing hardware at ARC consists of a DEC MicroVAX and a Symbolics Lisp workstation.

The Information Sciences Division at ARC is responsible for the development, in association with JSC, of TEXSYS. TEXSYS will control and monitor one configuration of the thermal test bed and will participate in the 1988 SADP single-system demonstration, as well as later SADP demonstrations. TEXSYS uses model- and rule-based reasoning under uncertainty, and exhibits fault detection, isolation, and recovery (FDIR) and limited executive control capability in the control and monitoring of the thermal test bed. Two knowledge-based tools, developed at ARC, were used to implement TEXSYS: Model Tool Kit (MTK) and Executive Tool Kit (XTK). MTK supports model-based reasoning, it can evaluate candidate faults through causal chains to the level of sensor or actuator failures, and states, time histories, and behavioral predictions are symbolically represented. XTK provides executive control functions. Both of these tools were built on the Symbolics workstation on top of Intellicorp's Knowledge Engineering Environment (KEE) expert system shell.

Figure 2-7 shows the TEXSYS architecture. The user interface is called the Human Interface to TEXSYS (HITEX). HITEX includes graphics and text display, including explanation and procedure rationalization information. HITEX was developed through interviews with thermal engineers and it was designed to be flexible and relevant to their needs. HITEX resides on a Symbolics Lisp workstation.

The Data Acquisition and Control System (DACS) is a conventional controller developed at JSC. TEXSYS and HITEX are connected to DACS via an Ethernet LAN. The interface between DACS and the expert system software is called the TEXSYS Data Acquisition System (TDAS). DACS run on a

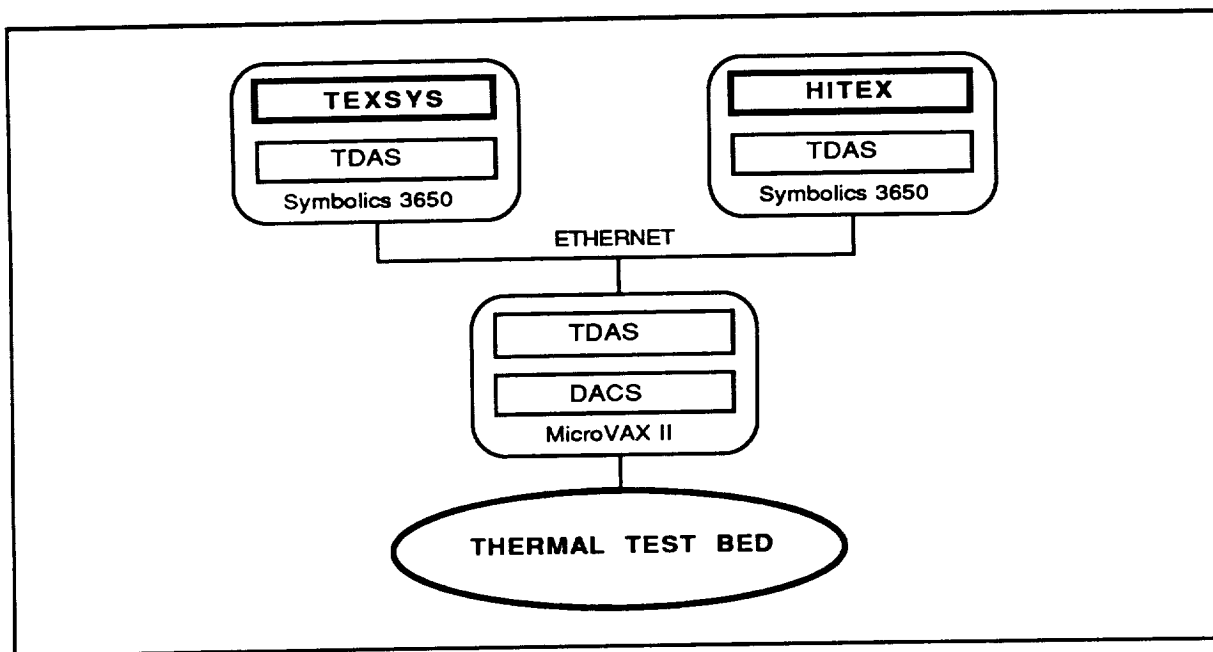


Figure 2-7. TEXSYS Architecture
(Source: Wong et al., 1988)

DEC MicroVAX II. TDAS runs on the MicroVAX and the Symbolics machines. TDAS acquires data from DACS, and only passes important data on to TEXSYS when these data change. TDAS also passes TEXSYS commands to DACS.

Verification and Validation. To test TEXSYS software, test plans have been generated which include conventional software testing methods as well as simulation and fault injection methods. Faults can be injected as fake sensor data into either simulated or real data to allow the evaluation of the TEXSYS software in response to particular faults.

Integration. Testing of the TEXSYS software is scheduled to be completed by September, 1988. Subsequently, TEXSYS will be shipped to JSC for integration with DACS. JSC will be responsible for TEXSYS maintenance and further development after the 1988 SADP demonstration.

2.1.4 Electrical Power System Test Beds

The Electrical Power System (EPS) will provide, distribute, and store electrical power for the Space Station Freedom systems and elements. The end-to-end EPS architecture begins with the collection of solar energy and extends to but does not include the actual loads or the power supplies within those loads every place electric power is consumed onboard the manned base. The baseline Space Station Freedom uses photovoltaic sources to convert solar energy to electric energy (SSP, 1988^c).

The Lewis Research Center (LeRC) is responsible for the development of the EPS, that is, the part of the power system which *generates* the electrical power. Marshall Space Flight Center (MSFC) is responsible for developing the Space Station Freedom Module (SSM)/Power Management and Distribution System (PMAD), that is, the part of the power system which *distributes* the power within a Space Station Freedom module. Each of these centers has developed an electrical power test bed related to their respective responsibilities: generation or distribution of power. In addition, JSC has developed the

Generic Electrical Power Distribution and Control (GEPDC) test bed, working with LeRC to define DMS-EPS interfaces.

2.1.4.1 LeRC Automated EPS Test Bed

LeRC Power System Engineering Division's EPS Test Bed capabilities include a physical power system test bed, a SSFP Software Laboratory, and an SADP software laboratory. A separate building being built at LeRC for a new Station Power Systems Facility will house the physical test bed and SSFP Software Laboratory. It is anticipated that this facility will also house the SADP software laboratory.

Physical Test Bed. The LeRC Automated Space Station Freedom EPS Test Bed, shown in Figure 2-8, is a 25 kW 20 kHz two channel ring bus system which is capable of supplying simulated solar dynamic power on one channel and photovoltaic power on the other. This power is converted to 440 VAC 20 kHz and fed to a pair of Power Distribution and Control Units (PDCUs) through a Main Bus Switching Assembly (MBSA). The power is distributed at 208 VAC to a load network through appropriate Load Converters which supply necessary load power characteristics (Kish et al., 1988). The hardware consists of prototype boards and switches, larger in size than future flight hardware, built by Westinghouse for Rockwell International Corporation's Rocketdyne Division, the prime EPS contractor.

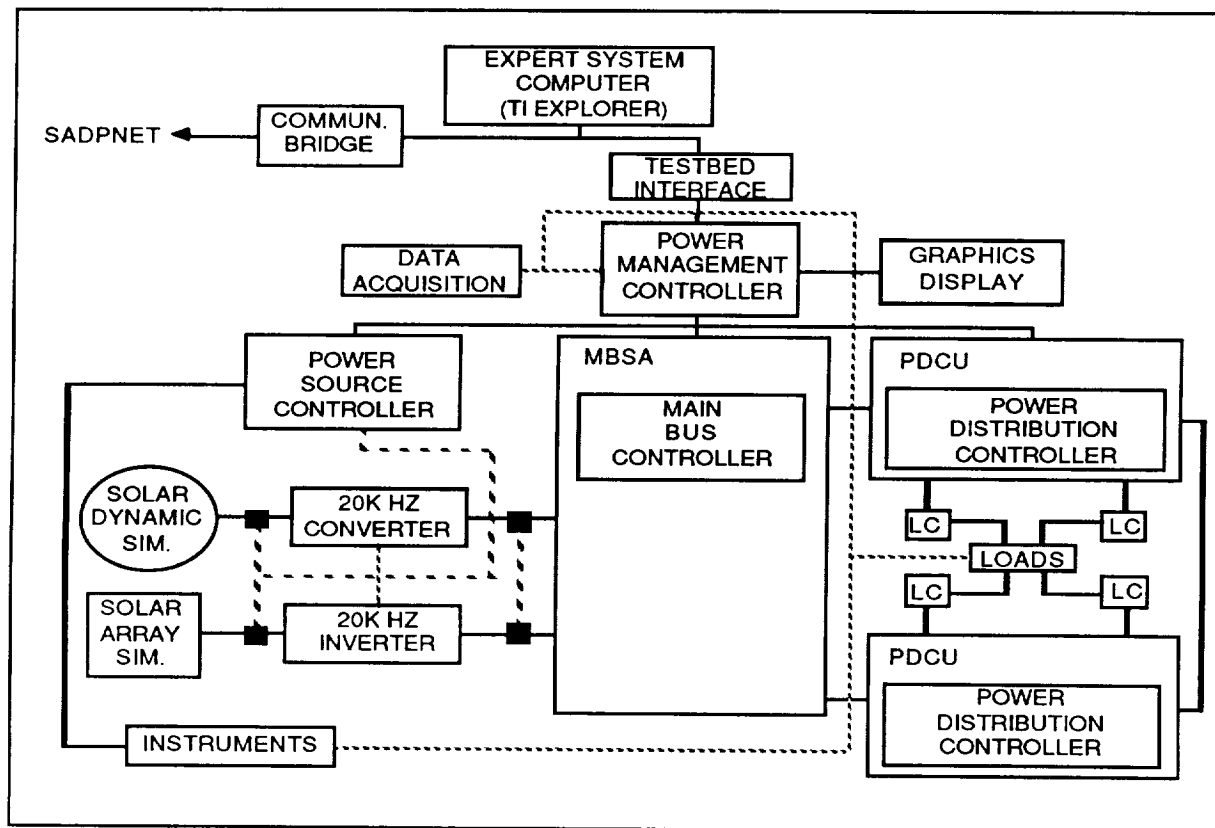


Figure 2-8. LeRC Automated Space Station Freedom EPS Test Bed
(Source: Kish et al, 1988)

The power management control scheme is hierarchical. The local dedicated controllers for source, main bus, and power distribution are Intel 80286 microprocessors and communicate with the power element smart interfaces through a MIL 1553b control link. The top level Intel Power Management Controller oversees lower level controllers on an Ethernet-like link. Additional data acquisition and real-time test bed graphics display terminals are connected to the Power Management Controller via serial and Direct Memory Access links.

The LeRC EPS Test Bed has three different power sources available:

- Solar Array Simulator, max output of 50 kW at 200VDC
- Solar Dynamic Simulator, up to 15 kW at 1200 Hz
- 20 kHz power supply for bypassing the simulators

At this time, the test bed has no energy storage capabilities. The addition of energy storage devices is planned.

SSFP Software Laboratory. The LeRC SSFP Software Laboratory is funded by SSFP and controlled by the LeRC Power Systems Engineering Division of the Space Station Systems Directorate. This laboratory is developing power system simulation software and comprises the following hardware:

- DEC VAX 11/785
- Applied Dynamics International (ADI) AD100
- Texas Instruments (TI) Explorer II-LX
- Apollo workstations
- IBM or IBM-compatible XT and AT microcomputers

The DEC VAX 11/785 serves as a host for the other processors listed above. The ADI AD100 is a high-speed machine capable of handling large amounts of Input/Output and Analog-to-Digital/Digital-to-Analog conversion. It is used to run a systems level model of the EPS network (e.g., flow of currents, states, fault simulation, topology changes). The TI Explorer II-LX is a dual-processor computer comprising a Motorola 68020 numeric processor and a separate Lisp processor. The numeric processor runs the UNIX operating system. The Lisp processor currently runs Inference Corporation's Automated Reasoning Tool (ART). This machine will also run Intellicorp's KEE in the near future.

SADP Software Laboratory. The LeRC SADP software laboratory is funded by OAST and is controlled by the LeRC Power Systems Technology Division within the Aerospace Technology Directorate at LeRC. This laboratory is developing power system fault detection and system monitoring software for the upcoming SADP demonstrations and comprises the following hardware:

- Texas Instruments (TI) Explorer II-LX
- DEC Micro VAX
- Apollo DN4000 minicomputer
- Ethernet network

The TI Explorer runs the UNIX operating system with KEE. The Ethernet network will connect the Power Systems Technology Division computers with the EPS Test Bed and the Station Software Laboratory.

The SADP Laboratory is utilizing a single PDCU identical to those on the EPS Test Bed, and an EPS simulator for local testing of interfaces (e.g., MIL 1553 interface for data acquisition) and for development of fault detection and system monitoring software.

Advanced Automation. Automating the operation of the Space Station Freedom's EPS falls into two broad categories: apportioning the available resources (electrical power) and operating the subsystems. The energy apportionment tasks address supervision of the power system's loads, energy storage, and distribution networks. Automating these activities encompasses (1) forecasting the energy that is available over a fixed time period (planning horizon) for both nominal and partially failed operations; and (2) apportioning the forecasted energy among the storage system and the loads, reserving enough energy for periods of eclipse, to maximize experiment productivity, and to respond to emergency power losses. The task of operating the various subsystems requires (1) qualifying a subsystem's operating state and performance status; (2) proposing, evaluating, and selecting strategic goals; and (3) planning a sequence of activities (Dolce and Faymon, 1986^a).

A diagram of EPS automation concepts for the initial Space Station Freedom is shown in Figure 2-9. LeRC's automation strategy is a three-staged approach:

- First, given the qualitative symptoms produced by a performance status program which uses conventional state estimation information, a rule-based expert system would diagnose subsystem failures. The diagnosis program would use fault-symptom association rules to determine the probable cause of the symptoms

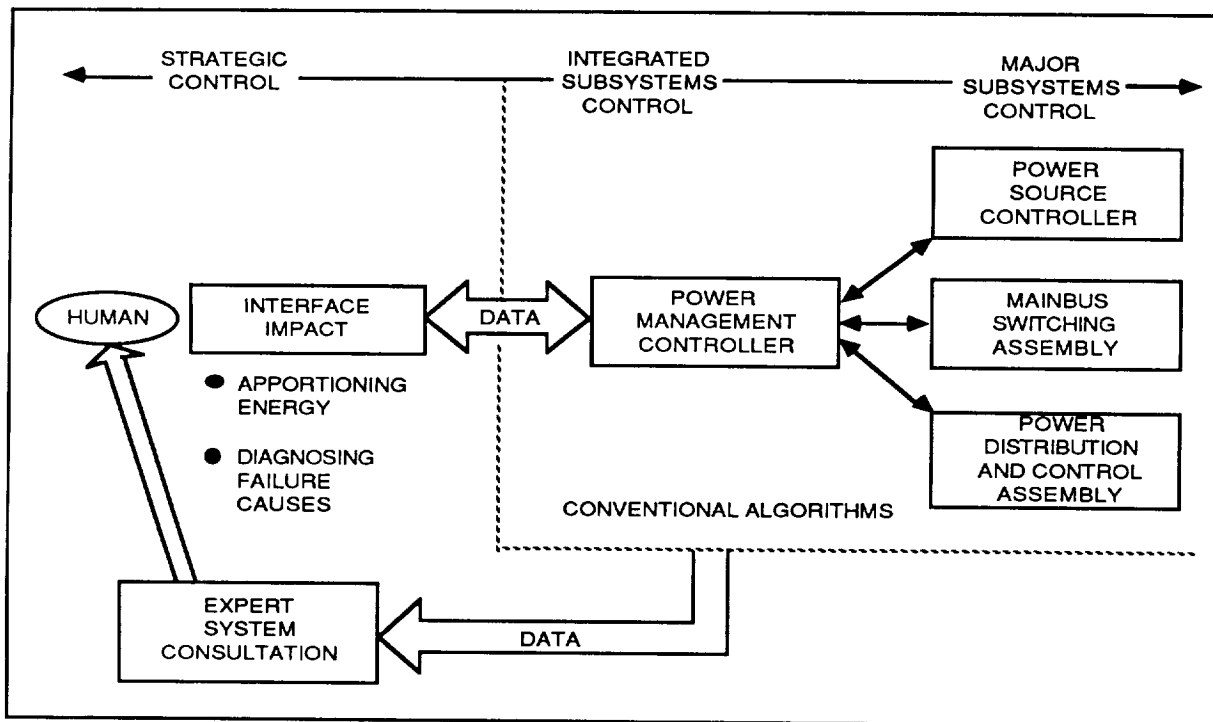


Figure 2-9. Power System Automation (Initial Space Station Freedom)
(Source: Dolce and Faymon, 1987^c)

- Secondly, a program that reasons using knowledge of a system's function--perhaps from a simple model--would determine a cause, should the first approach fail
- A third program would propose failure cause hypotheses and verify them by experimenting with the actual system, if the second program failed

The probable cause of failure would then be used in selecting appropriate strategic goals and in revising the forecasts of available energy. These concepts are developed further in (Dolce, 1987a; Dolce, 1987b; and Dolce and Faymon, 1987).

The LeRC knowledge-based system, collectively referred to as the Power Management and Control System (PMACS), includes the Automated Power Expert (APEX), an associated graphical operator interface known as Human Interface to Power (HIP), and the necessary hardware and software interfaces to the Space Station Freedom EPS Test Bed control system. APEX, shown in Figure 2-10, consists of a fault detection/diagnostic system and associated knowledge base residing on the Lisp environment of a TI Explorer II/LX workstation. A load planner/scheduler is being developed in the C language and will execute on the TI UNIX processor, with communications via a remote procedure call capability within the system software. The knowledge base consists of the diagnostic/scheduling rule base and real-time test bed fact base updated through an Ethernet communications module. Initial fault system prototyping is being done with KEE 3.1 (Kish et al., 1988).

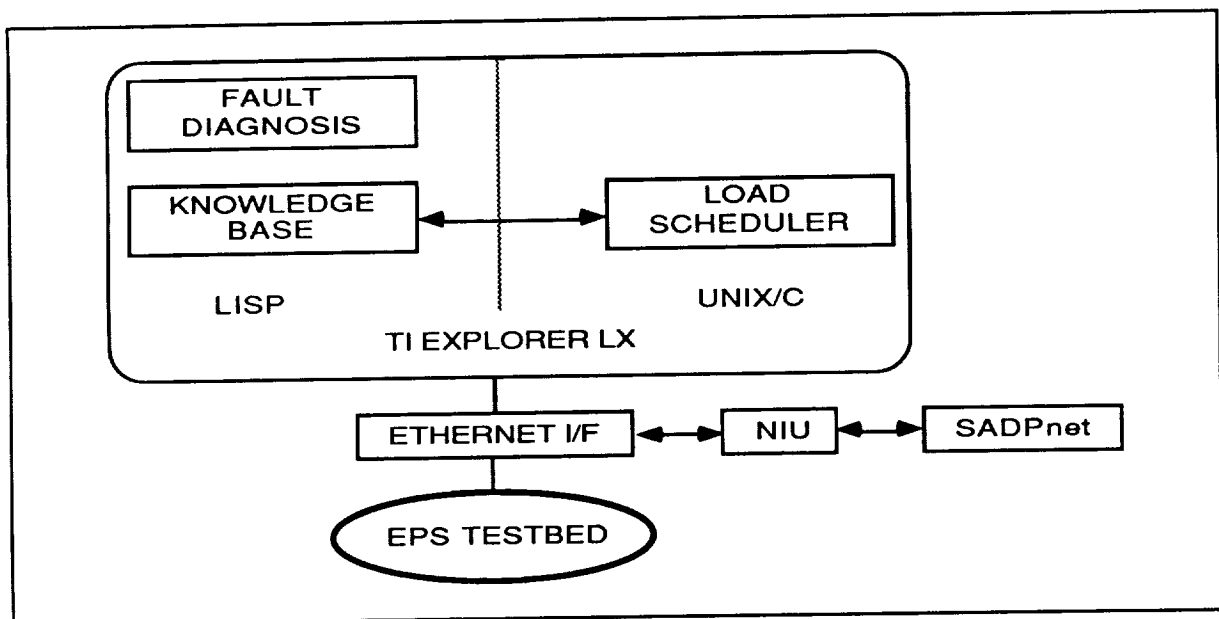


Figure 2-10. LeRC EPS APEX Automation Concept
(Source: Kish et al., 1988)

Other capabilities which will be added to the APEX software functionality include system health monitoring and trend analysis, which is important for incipient fault condition detection. The EPS knowledge-based fault system will not only be heuristic, but incorporate causal modeling. System components and their attributes, including functional, behavioral, and interactive characteristics, will be represented within the knowledge base in a structured fashion. The behavioral properties must describe current state information as well as single or multiple fault expectations. These underlying relations, which

are often lost or masked in typical rule-based systems, need to be incorporated within the expert systems (Kish et al., 1988).

JSC's Mission Planning and Analysis Division (MPAD) is developing, with the assistance of Inference Corporation, an Ada version of ART. Once this software becomes available, LeRC will be a Beta test site and will then be able to implement KEE, ART and AdaART versions of the same application software for comparison.

Integration. The schedulers being developed at LeRC concentrate only on EPS constraints and are not coordinated with the scheduling of other resources such as thermal control, communications bandwidth, crew time, etc. Parallel schedulers for each Space Station Freedom system, each with its own constraint set and conflict resolution strategies, will be required. LeRC feels the largest Space Station Freedom development issue will be this type of cooperative problem solving, that this is an OMS responsibility, but are uncertain as to how OMS is working to solve this problem. LeRC was not aware of an established mechanism for two-way communication with OMS developers.

Verification and Validation. No formal Verification and Validation (V&V) plans exist. Informal V&V of the knowledge-based system software within the LeRC SS EPS Test Bed will consist of interactive simulation to determine if the system performs correctly, that the software does not jeopardize the hardware, and that the test bed is robust. This simulation may not execute in real time.

2.1.4.2 MSFC Power Management and Distribution System Test Bed

The Electrical Power Division of the Science and Engineering Directorate of MSFC is responsible for developing the Space Station Freedom Module (SSM)/Power Management and Distribution System (PMAD); that is, the part of the power system which *distributes* the power within a Space Station Freedom module. The SSM/PMAD is part of the MSFC Autonomously Managed Power Systems Laboratory (AMPSLAB).

Space Station Freedom modules are the physical Space Station Freedom structures in which the crew of the manned base will live and work; two examples are the U.S. habitation and laboratory modules. The SSM/PMAD will accept 20 kHz power from the station EPS and distribute it to the various loads inside these modules. These loads include the Station subsystem loads such as the pumps for the Environmental Control and Life Support System (ECLSS), as well as the experimental payloads and crew tools, including recreational facilities (Kish et al., 1988).

The Space Systems Company of the Martin Marietta Astronautics Group in Denver, Colorado, has been working with MSFC for the past three years in defining and developing an advanced development SSM/PMAD test bed to advance the state-of-the-art in space power systems power management and distribution (Weeks, 1988). This test bed will function as a highly intelligent load on the LeRC Space Station Freedom EPS Test Bed for the 1990 SADP demonstration. An SSM/PMAD human interface module will be developed for the 1990 demonstration and will provide an intelligent interface to PMACS data from LeRC (Kish et al., 1988).

Physical Test Bed. The SSM/PMAD Test Bed, shown in Figure 2-11, employs single-phase 208 volt, 20 kHz electrical power and is large enough to operate a substantial number of realistically sized loads simultaneously and autonomously (Freeman et al., 1988). Autonomy has been embedded down to the lowest level processors (LLPs). These processors are 68010-based systems with each LLP managing its load center, subsystem distributor, or PDCU. Load centers distribute electrical power to equipment racks. Subsystem distributors distribute electrical power to separable subsystems. A VME/10 minicomputer (the communications and algorithmic controller) controls the test bed communications (Ethernet, and RS-232 and RS-422 serial data networks) and directs the LLPs (Kish et al., 1988). Software for the LLPs and the VME10 is written in Pascal.

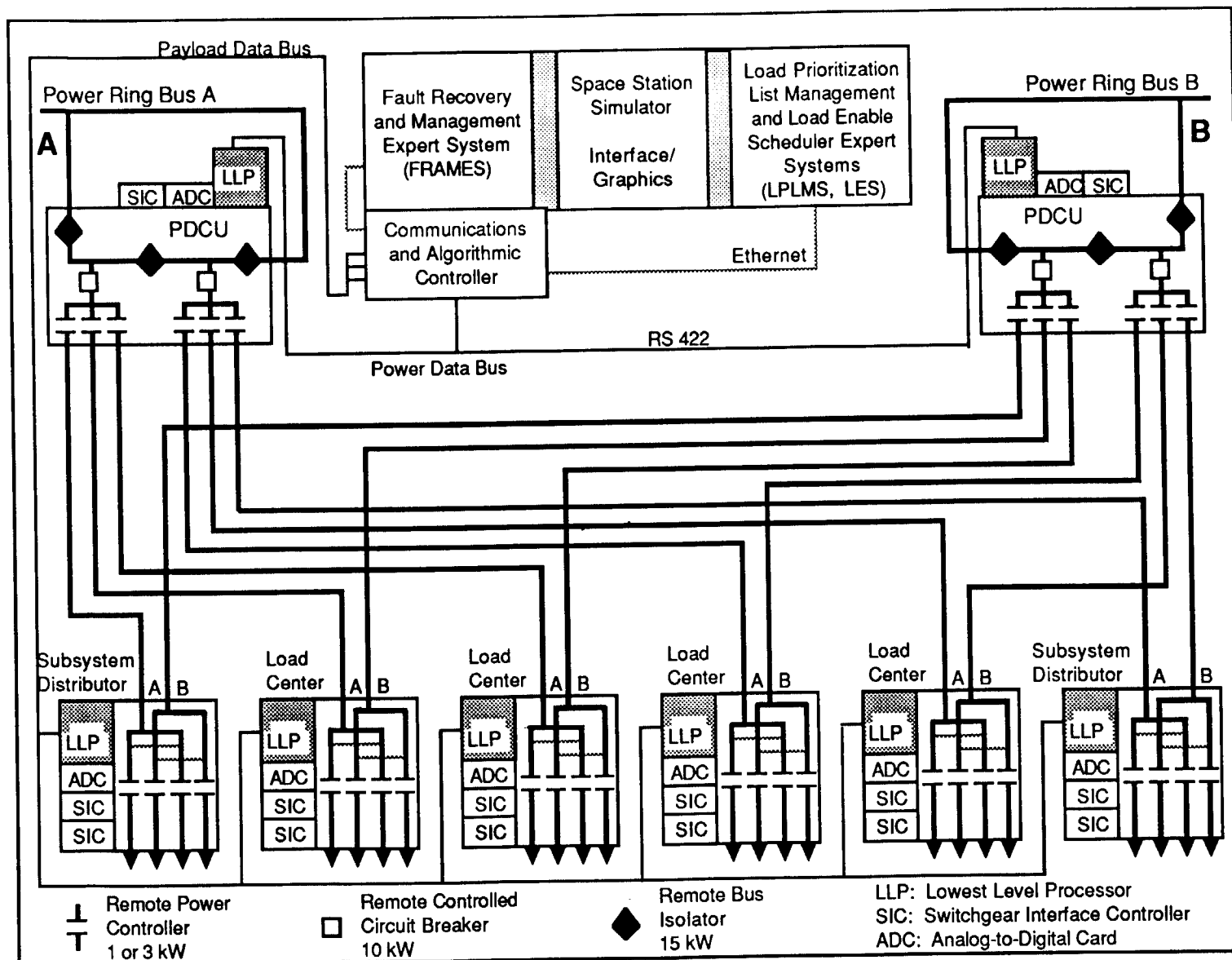


Figure 2-11. MSFC SSM/PMAD Test Bed
(Source: Lollar and Weeks, 1988)

The SSM/PMAD Test Bed includes the following computing hardware:

- VME 10 minicomputer
- Xerox 1186 workstation
- Symbolics 3620 workstation

Separate copies of the SSM/PMAD are housed at both MSFC and Martin Marietta (Denver, Colorado).

Advanced Automation. The test bed features three cooperating knowledge-based systems, the Loads Priority List Management System (LPLMS), the Loads Enable Scheduler (LES), and the Fault Recovery and Management Expert System (FRAMES), as shown in Figure 2-12.

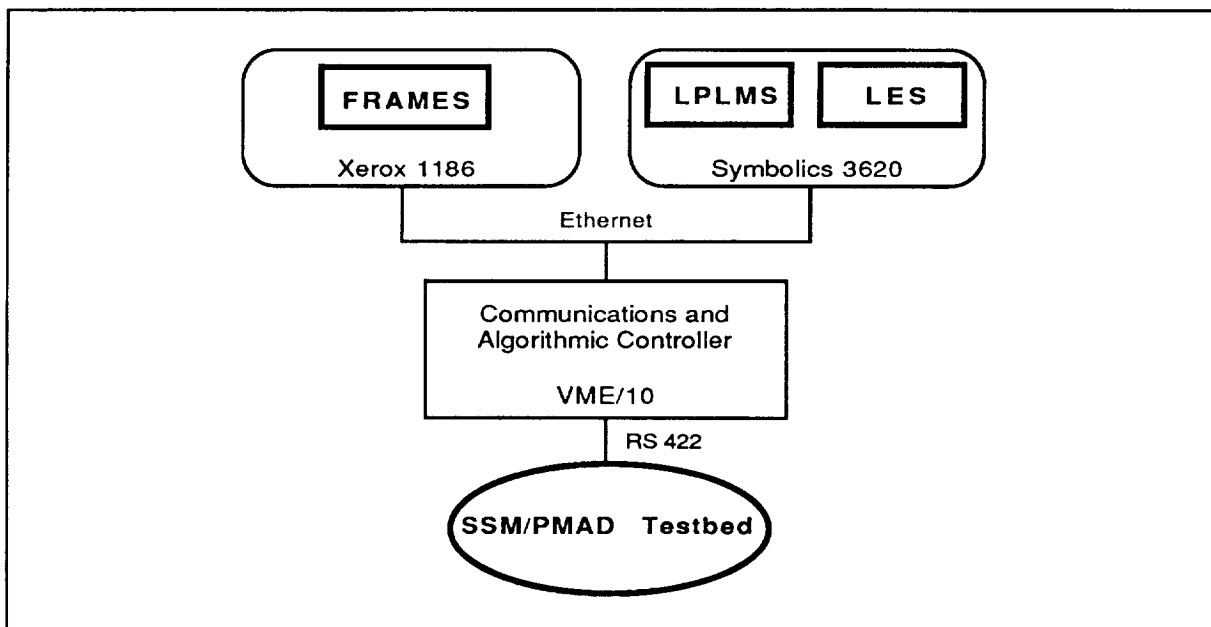


Figure 2-12. SSM/PMAD Automation Architecture
(Source: Kish et al., 1988)

The prioritization expert system is called the LPLMS. The LPLMS keeps up with the dynamic priorities of all payloads while developing current load shedding lists for the LLPs every 15 minutes in preparation for contingencies which necessitate load shedding (Lollar and Weeks, 1988). Load priorities often change. As a resource nears depletion (and, therefore, possibly becomes more mission critical), its priority may change. For example, 60 day non-interruptible crystal growing experiment may have a different priority on day 59 than it had on day 2. LPLMS ensures that the system will never erroneously shed a critical payload.

The communications and algorithmic controller develops individual load center load shedding lists from the LPLMS list and downloads the information to the LLPs every 15 minutes. After quickly developing a load shedding list, the LPLMS then refines the list in a background task. The LPLMS is object ori-

ented, uses heuristic search, and uses inputs to alter the way it runs. LPLMS is implemented in Common Lisp and resides on a Symbolics Lisp workstation (Kish et al., 1988).

The scheduler expert system is called the Load Enable Scheduler (LES). LES schedules and reschedules the payloads for the SSM/PMAD test bed. LES is actually a special version of MAESTRO, the Martin Marietta resource scheduler. LES uses several AI technologies, including object-oriented programming, heuristically guided search, activity library, expert functions, etc. LES schedules loads with regard to numerous resource constraints such as available crew members, supplies of payloads, interdependence of payloads, power profiles, thermal status, etc. LES is also written in Common Lisp and resides on a Symbolics workstation (Kish et al., 1988).

The fault recovery expert system is called the Fault Recovery and Management Expert System (FRAMES). FRAMES watches over the entire SSM/PMAD test bed operation looking for anomalies and impending failures. The functionality of FRAMES actually extends to the lowest level processors in the test bed for comprehensive fault management of the test bed. FRAMES is responsible for detecting faults, advising the operator of appropriate corrective actions, and within the load centers and subsystem distributors, autonomously implements corrective actions through power system reconfiguration. The system will carry out trends analysis seeking incipient failures and soft shorts as well as open circuits. FRAMES will handle soft and masked faults in the switchgear, the cables, and the sensors. Hard faults will be handled quickly by the switchgear to protect the PMAD system. FRAMES is implemented in the Common Lisp Object System (CLOS) and resides on a Xerox 1186 AI workstation (Kish et al., 1988).

The front end to LES contains interfaces to LES and to the LPLMS and the user interface. The test bed is capable of running in an autonomous closed-loop mode. When a fault or anomaly occurs in the SSM/PMAD test bed, FRAMES detects, diagnoses, and recommends corrective action (automatically implementing corrective actions where appropriate). Following corrective actions, LES determines if a new payload schedule is necessary and, if so, reschedules the loads in accordance with the new configuration of the SSM/PMAD. The LPLMS derives a new load shedding list from the new schedule and forwards it to the communications and algorithmic controller which breaks the new load shedding list down into individual load center and subsystem distribution load shedding lists. These local load shedding lists are then downloaded into the appropriate LLPs (Kish et al., 1988).

SSM/PMAD Status. The SSM/PMAD breadboard is still being completed at the time of this writing. The scheduler, LES, is completed as is the LPLMS prioritization expert system. FRAMES is almost completed with some final coding to be performed. The conventional software is essentially finished. The next task is the overall integration and testing. The SSM/PMAD has been selected by NASA to be a participating test bed in the SADP 1990 demonstration.

Integration. As stated above, MAESTRO, the scheduler used within LES, is proprietary to Martin Marietta. It can take into account numerous constraints external to the power system. Part of the scheduling functions performed by LES could be considered to be OMS responsibility. As with the LeRC scheduler discussed above, MSFC personnel are not aware of current plans for the OMS to provide these functions, nor were they aware of the proper mechanism for obtaining this information or for providing information to the OMS development efforts.

Verification and Validation. The MSFC SSM/PMAD contact was uncertain regarding plans for V&V of the knowledge-based system software. Any testing probably will be informal, ad hoc, trial and error testing using test cases.

Other Issues. Some concern was expressed about hooks and scars for advanced automation within the SSFP plans. A constrained Space Station Freedom budget may reduce knowledge-based system development for use on the manned base. The fear is that knowledge-based system applications

may not be developed for Space Station Freedom unless hooks and scars for their evolutionary inclusion are worked into SSFP plans, especially in the Program Definition and Requirements Document (PDRD).

2.1.4.3 JSC Generic Electrical Power Distribution and Control Test Bed

The Generic Electrical Power Distribution and Control (GEPDC) Test Bed, shown in Figure 2-13, provides a simulation of the Space Station Freedom power generation and distribution system to evaluate candidate solutions to the Space Station Freedom power system. The GEPDC was developed by the Power Systems Group within the Avionics Systems Division at JSC, working with LeRC to define DMS-EPS interfaces, in support of the ETC integration efforts by providing a simulated electrical power system. A scaled 75 kW of 20 kHz power can be simulated through the use of an Intel-based breadboard Power Management Controller (PMC), functional simulation, and math models. Nominal electrical power supplied is 90% to 133% of the 75 kW with the 133% power (100 kW) available for eight hours out of every 24 hour period. The GEPDC breadboard 4 KVA system scales the power to 100 kW simulated levels. Two breadboard DC power supplies emulate two of the four Space Station Freedom Photovoltaic generation modules. Intel processors simulate the battery energy storage and load profiles. Control inputs and failures are simulated or induced by the operator of the Intel processors. The standard DMS ADS/DADS DMS systems services are used to communicate with other nodes on the DMS LAN (Ulmer, 1988).

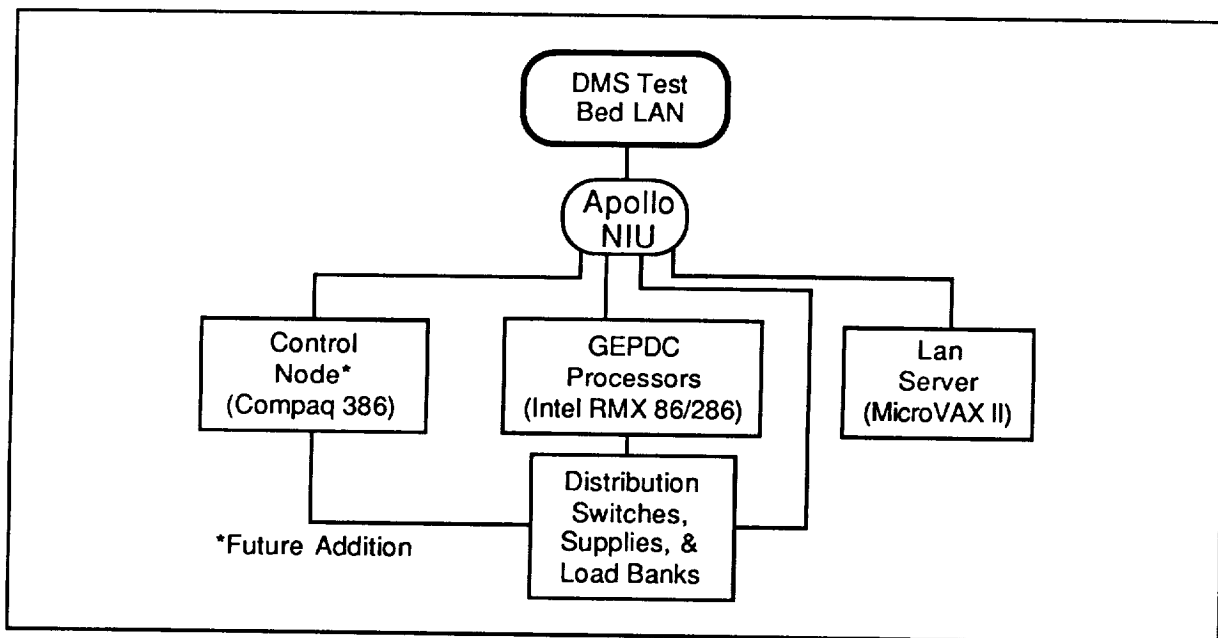


Figure 2-13. Generic Electrical Power Distribution and Control Test Bed
(Source: Ulmer, 1988)

2.1.5 JSC Guidance, Navigation, and Control Emulator Test Bed

JSC's Guidance, Navigation, and Control System (GN&C) is divided into the following two major subsystems: core GN&C subsystem and GN&C traffic management subsystem. The core GN&C subsystem provides attitude state maintenance and orbital state maintenance of the Space Station Freedom. In addition, the core GN&C subsystem supports the pointing of the power system and

and provides Space Station Freedom state and attitude information to other systems' and users. The GN&C traffic management subsystem is responsible for controlling incoming, outgoing, and station-keeping traffic within the Command and Control Zone (CCZ) of Space Station Freedom; controlling docking and berthing operations; monitoring the trajectories of vehicles and objects which may intersect the orbit of Space Station Freedom; predicting potential collisions; and supporting flight planning of the traffic around the Space Station Freedom (SSP, 1988^d).

The Avionics Systems Division's GN&C Emulator Test Bed allows the evaluation of different types of data bus architectures, the evaluation of SDP prototypes, the evaluation of the use of Ada in a distributed environment, and integration with other test beds. The test bed consists of a set of small computers connected with communication busses, emulating the distributed architecture, software, programs, and other functions of the GN&C System. Figure 2-14 shows the configuration of the GN&C Emulator Test Bed. Physical hardware consists of the following computers and networking equipment:

- 5 Compaq Desk Pro 286 PCs (IBM AT Class)
- 5 Compaq Desk Pro PCs (IBM XT Class)
- VME System (Dual Motorola MC 68020 processors)
- DEC MicroVAX II Lab Series Processor
- HP 1000 Series F Computer
- Dual IEEE-488 Spec LANs

The MicroVAX will figure into future integrated testing when SDP functions are moved to it. The HP 1000 computer is used to support test data archiving and post processing. The dual LANs operate independently.

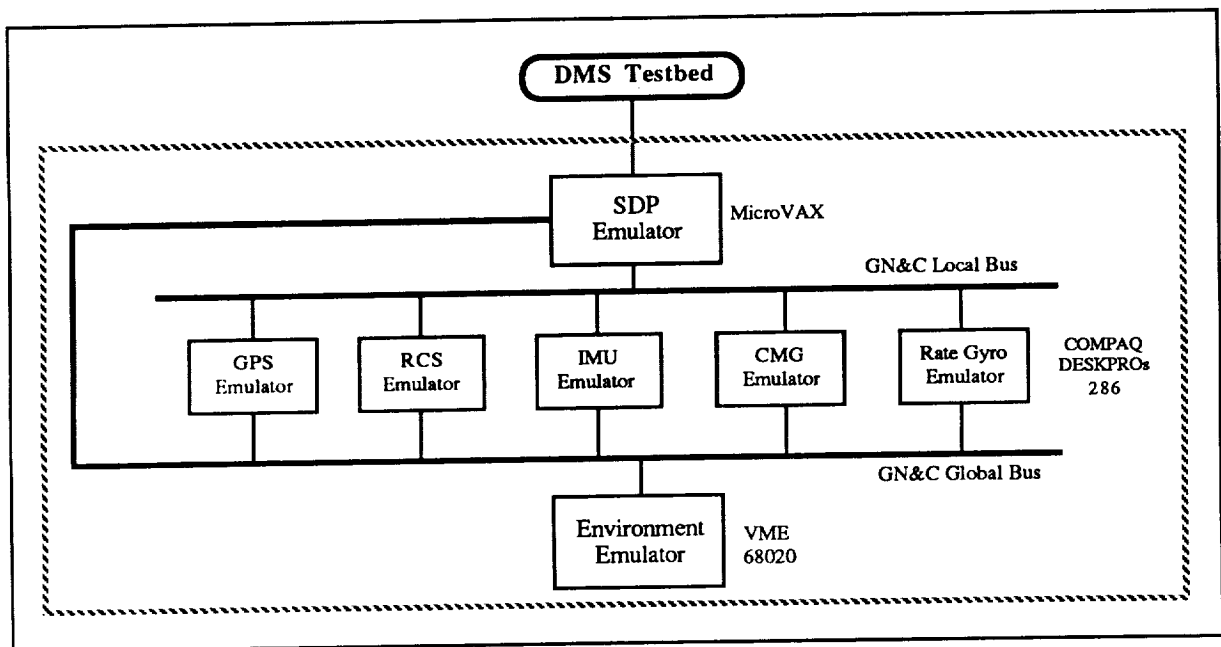


Figure 2-14. GN&C Emulator Test Bed
(Source: Ulmer, 1988)

The GN&C hardware, including sensors and effectors, is currently emulated using math models resident in individual microprocessors (Compaq Desk Pros). As the test bed evolves, these models will be replaced with prototype flight system hardware (e.g., SDPs) as it becomes available. One of the test bed computers, the On Board Check Out (OBCO) computer, stores the Space Station Freedom equations of motion, mass properties computations, environment equations, and simulation control provisions. The OBCO is connected to each of the other computers through a separate data bus. The test configuration, therefore, has three separate and distinct data busses: the GN&C local bus, the OBCO bus, and a subset of the DMS global bus (Ulmer, 1988).

Advanced Automation. The OBCO application is a prototype of the OMS testing function and is divided into three categories: Orbital Replacement Unit (ORU) level testing and checkout, intra-system testing, and inter-system testing.

Integration. The first two OBCO categories will be demonstrated during Phase II of the ETC Integrated Testing (see Section 2.2.1) using the OBCO application in concert with the GN&C simulated flight hardware. During these tests, redundant hardware will be activated and brought up to speed from a *cold standby* mode. Functional tests will be completed before this hardware will be allowed to process flight critical data. These tests will include *wake up* self tests, mechanical functional tests, and interface communication tests. In addition, the Reaction Control System (RCS) will be checked out with communication functionality tests and hot fire tests of the simulated thrusters scheduled for reboost (Ulmer, 1988).

2.1.6 JSC Communications and Tracking System Test Bed

JSC's Communications and Tracking System (C&T) provides all Space Station Freedom manned base communications services including audio, video, space-to-space communications, and space-to-ground communications. The C&T system also provides the necessary tracking services to GN&C. The system has been divided into six subsystems, each representing a major class of service or function.

The space-to-space subsystem provides all communications with the NSTS, Extra-Vehicular Activity (EVA), Orbital Maneuvering Vehicle (OMV), Mobile Servicing Center (MSC), and other future Space Station Freedom Program Elements (SSFPEs). The space-to-ground subsystem provides communication via the Tracking and Data Relay Satellite System (TDRSS) to the ground data networks. The audio subsystem provides all the internal and external voice communication for the Space Station Freedom.

The video subsystem provides all the internal and external video for the Space Station Freedom. The tracking subsystem consists of a Global Positioning System (GPS) receiver/processor with provisions to accommodate laser docking and radar in the future. The control and monitor subsystem provides for the management of all the C&T resources as well as the C&T data distribution function (SSP, 1988^a).

The C&M Test Bed is used to develop and evaluate candidate software for the Space Station Freedom C&T system. The configuration of the test bed is shown in Figure 2-15 (Ulmer, 1988).

Physical Test Bed. The test bed consists of (at least) the following hardware:

- Symbolics 3670 Lisp workstation
- SUN 3/260 workstation
- DEC VAX 11/750
- DEX MicroVAX II
- Apollo NIU

- Local Ethernet LAN

Software. The software components of the test bed consist of a Mass Storage Unit Configuration and Status Data Base, residing on the VAX 11/750 and Subsystem and Space-to-Ground simulators residing on the MicroVAX, as well as two expert systems.

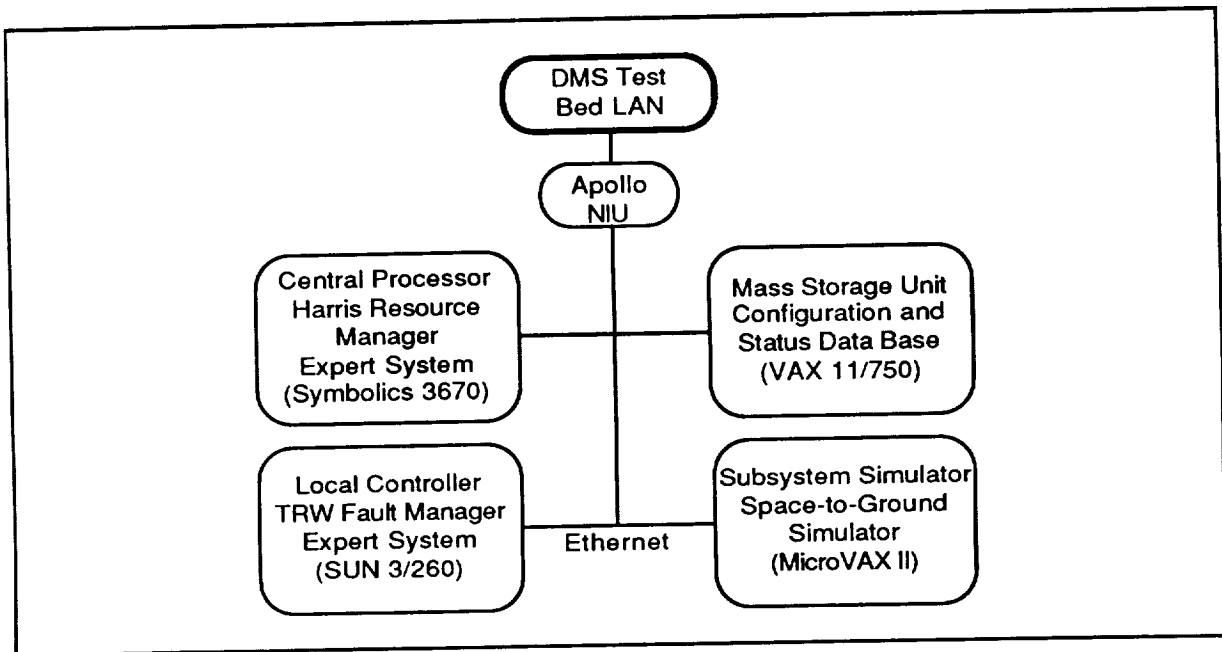


Figure 2-15. C&T Control and Monitoring Test Bed
(Source: Ulmer, 1988)

Advanced Automation. A Local Controller Fault Manager Expert System, built by TRW, was implemented on the Sun 3/260 workstation. A Central Processor Resource Manager, built by Harris, was implemented on the Symbolics 3670 using ART. These systems are currently being converted to C and Ada to facilitate their use on the targeted flight processor environment (i.e., Intel 80386) and the UNIX operating system (Ulmer, 1988).

Integration. The C&M Test Bed is interfaced with the DMS Test Bed network. Integrated tests are planned to demonstrate the interaction between the C&T C&M Test Bed, OMS, GN&C, and the SSCC. At a later date, the prime C&T subcontractor, RCA, will test flight applications software in the C&M Test Bed using SDP engineering models.

2.1.7 ARC Advanced Architecture Test Bed

The Advanced Architecture Test Bed is managed by the Information Sciences Division at the NASA ARC. The goals of the test bed are to investigate the hardware and software issues on onboard multiprocessor systems applications to NASA missions, and to provide the required prototyping capability for transfer of the architecture technologies to specific NASA projects. Technical problems addressed by this test bed include the following:

- Real-time fault tolerant architectures for knowledge-based multiprocessor systems including dynamic system reconfiguration using on-chip components

- Management and control of large distributed knowledge data bases in excess of 10 gigabytes including dynamic memory allocation and reallocation
- Operating systems for a distributed heterogeneous environment
- Validated compilers and data translators for transition from a development environment to an Ada-based operational environment optimized for real-time system performance
- Optimized run-time performance for radiation-hardened modules
- Automated load scheduler for efficient use of the multiprocessor environment
- Technology concepts for high speed memory access and data transfer time
- Definition and evaluation of hooks and scars guidelines for processor evolution and user transparency

Physical Test Bed. The Advanced Architecture Test Bed, shown in Figure 2-16 consists of two networks, fiber optic dual redundant counter rotating token-ring and Ethernet, connected by a level 3, TCP/IP gateway. The test bed includes advanced hardware and software technologies and concepts from academia (from collaborative research grants), industry (*beta* hardware and software, of which there are five participating contractors including TI, IBM, DEC, IIM, and Symbolics), and DARPA (Compact Lisp Machine) (Lum, 1988).

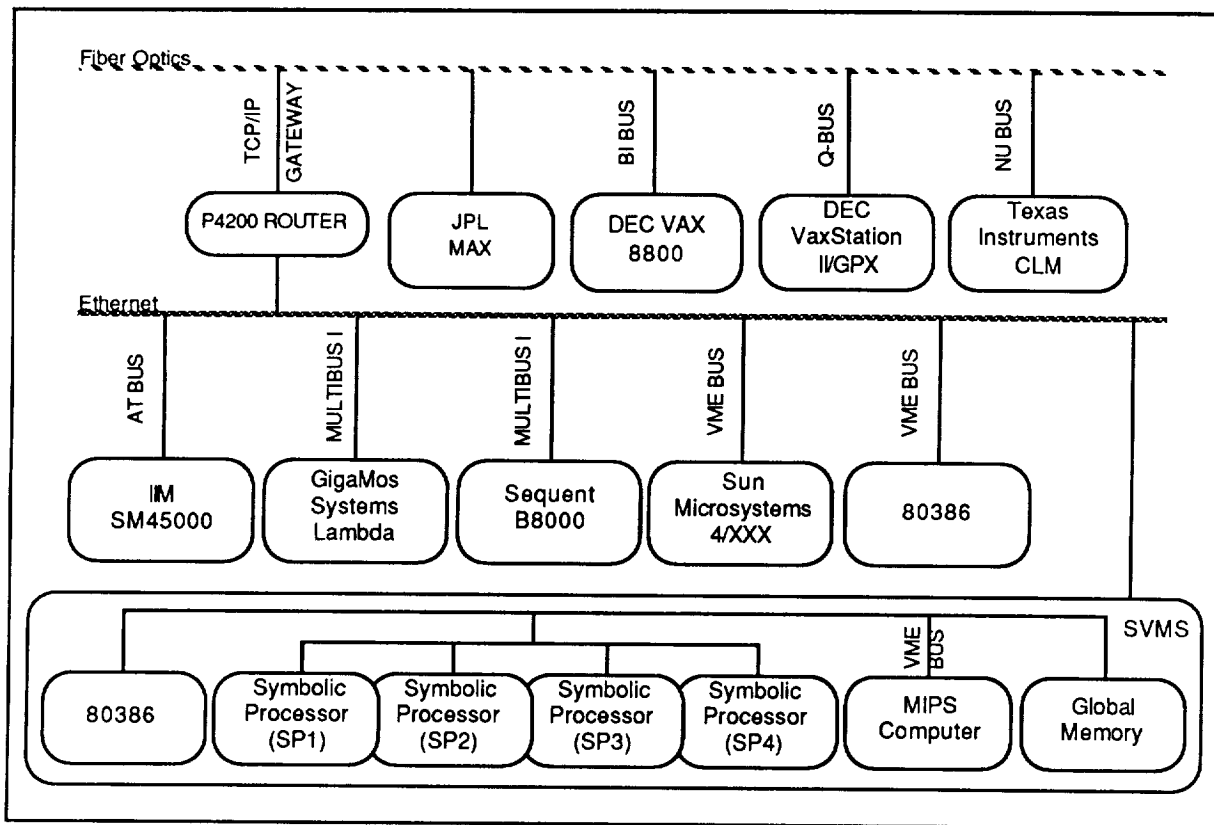


FIGURE 2-16. Advanced Architecture Test Bed
(Source: ARC, 1988)

This hardware includes a JPL MAX processor, a DEC VAX 8800, a DEC VAXStation II/GPX, a Texas Instruments Compact Lisp Machine (CLM), an Integrated Inference Machines (IIM) SM45000, a GigaMos Systems Lambda, a Sequent B8000, a Sun Microsystems workstation, an Intel 80386 processor, and an SVMS multiprocessor. The TCP/IP gateway is a Proteon Pronet-80 P4200 Router.

Simulation models for distributed multiprocessor systems including network configurations complement the hardware research test bed--these computer models run on the Symbolics 3675 and the DEC 8800. The results from the research test bed provide a cross-check on the contractor's development efforts during the three development phases for the Spaceborne VHSIC Multiprocessor System (described below) (Lum, 1988).

SVMS. The Spaceborne VHSIC (Very High Speed Integrated Circuit) Multiprocessor System (SVMS) is intended for real-time and non-real-time AI applications onboard the Space Station Freedom. The SVMS is being designed to overcome the limitations of numeric processors with respect to large, real-time, knowledge-based system. Its architecture includes a 32-bit numeric processor, four symbolic processors, and global memory. The objectives of the SVMS effort include the following:

- Self-testing, Self-maintaining, Automated Configuration
- Accommodation of one 32-bit numeric processor
- Accommodation of four 40-bit symbolic processors
- Optical interconnects
- VLSI/VHSIC technology and module compatibility
- Concurrent Common Lisp and Ada languages
- Minimum of 10 rads radiation resistance, no SEUs
- 25 MIPS sustained uniprocessor execution rate (target of 40 MIPS)

Phase I of the SVMS effort, scheduled for completion in April 1988, is the system feasibility effort including tradeoff analysis and development of computer simulation models. The Phase I team consists of TRW and Symbolics, Inc. who were selected through competitive procurement. Phase II, a 30-month effort scheduled for award during FY-88, is for the development of two competing brassboard systems containing flight-qualifiable components and the operating system environment. Phase III, a 48-month effort, is for the development of one flight-qualified system scheduled for delivery in early CY-95. One contractor from Phase II will be selected to do the Phase III effort (Lum, 1988).

LANES. The Local Area Network Extensible Simulator (LANES) provides a method for simulating the performance of local area networks. LANES incorporates two network models. One is a model of a FDDI ring network. The other is a Star*Bus. These models each simulate a unique link layer and share a common network and load layer. The simulation is designed to determine performance characteristics of FDDI and Star*Bus under a variety of loading conditions. It allows the user to objectively compare the performance of the two network architectures (LANES, 1987).

DPNS. The Distributed Processing Network Simulator (DPNS) allows the definition of a network architecture and simulates the running of a workload by that network to determine the effects of the network architecture on system performance. It allows workloads of arbitrary complexity to be defined and run on a model of the network's data processing resources. This will determine what type of response can be expected from those resources. DPNS is a distributed processing simulator. This means that resources utilized by a given job can be spread throughout a network. Centralized processing can also be modeled.

Integration. A major purpose of the Ames test bed is to investigate advanced computing concepts and to feed the results to JSC for possible use within the DMS Test Bed. Ames plans to transition to FDDI as soon as it is available. One of the SVMS brassboards will be delivered to JSC. The final design will not be tested for two years. However, Ames needs to keep current on the details of the DMS Test Bed design to ensure the SVMS-DMS interface is correct. Lack of adequate DMS design information could result in design and operational decisions that could cause compatibility problems and result in costly changes.

Important integration issues are the coordination of interface definitions between all test beds, communication, and common working agreements for technology transfer. The level at which the technologies are transferred, of course, defines the requirements for specifications, agreements, and communication. A major problem arises when software or hardware is delivered to one center from another if the required interfaces have not been defined. In one instance, a delivered prototype sat around for several months because the receiving center was not prepared to accept it and no money had been budgeted for the interface definition effort.

2.1.8 Other DMS Nodes

The Human Computer Interaction Node and the MPAC Displays and Control Node are participants in the Phase II ETC integration activities.

2.1.8.1 JSC Human Computer Interaction Node

The Manned Systems Division's Human Computer Interaction (HCI) Node of the JSC DMS Test Bed is shown in Figure 2-17 and is a prototype configuration which will allow the exploration of human factors issues through the presentation of information produced by the OMA and core systems nodes. The HCI Test Bed consists of a DEC MicroVAX running the VMS operating system.

The two major software components of the HCI are BLOX, a user interface management system (UIMS) used to build, maintain, and display graphical user interfaces, and the HCI Ada Executive (HAE) which uses DADS to access system data from other nodes and present it for use by BLOX programs.

For Phase II, the HCI prototype will be limited to monitoring systems. Command capabilities are planned when the systems and the network services are more mature (Ulmer, 1988).

2.1.8.2 JSC MPAC Displays and Control Node

The Systems Development and Simulation Division's MPAC Displays and Control (D&C) Node, shown in Figure 2-18, is used for the prototyping of the fixed MPAC hardware configuration and on-board crew controlling and monitoring activities in a Space Station Freedom environment.

Currently the MPAC D&C Node consists of a DEC MicroVAX II running the VMS operating system, a Raster Technology Model One/80 graphics terminal, and a Compaq 386 PC with a Tektronix 4107 terminal emulator. The display and control function will use resources generated by the Dataviews UIMS package. Communications over the DMS Test Bed network will use driver routines employing both the DADS and ADS services.

Initial applications of the MPAC D&C Node will be limited to monitoring Phase II operations. Subsequent enhancements will include a command generation capability in accordance with OMS Test Bed defined procedures (Ulmer, 1988).

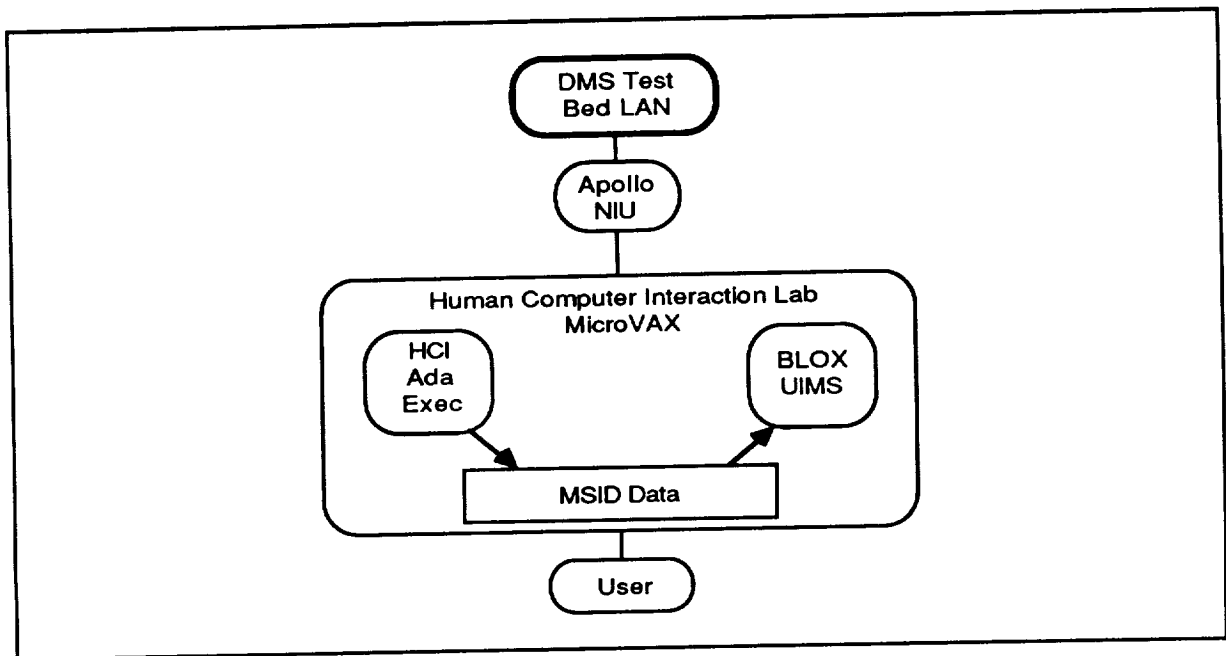


Figure 2-17. Human Computer Interaction (HCI) Node
(Source: Ulmer, 1988)

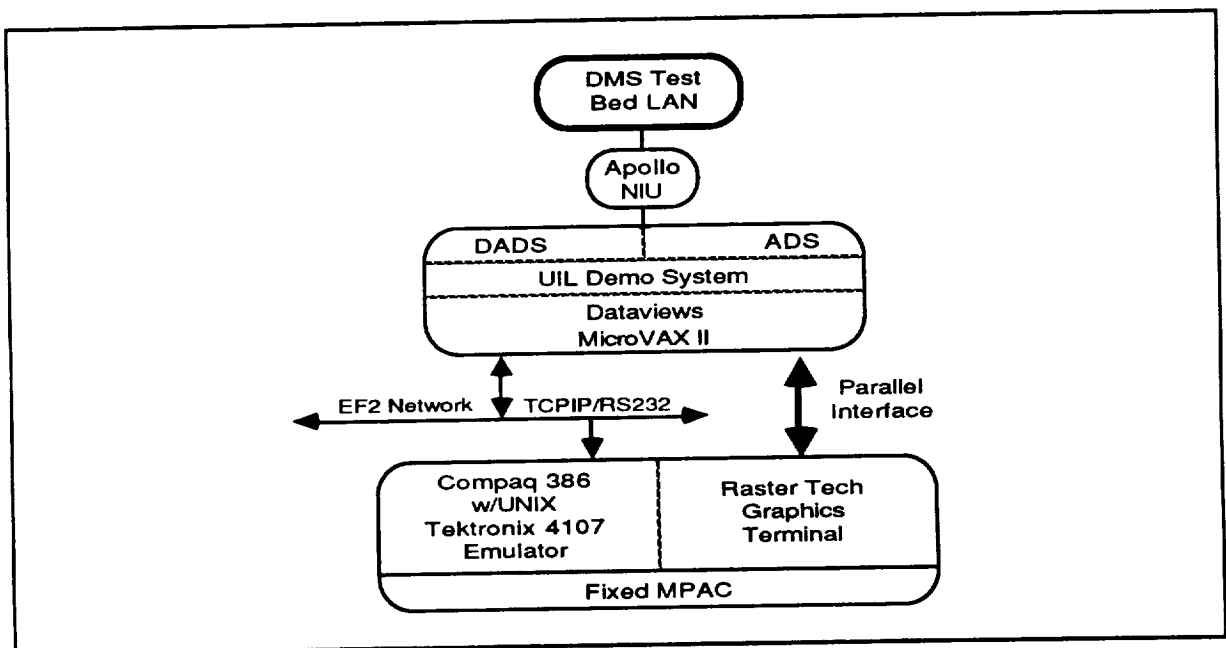


Figure 2-18. MPAC Display and Control Node
(Source: Ulmer, 1988)

2.2 MAJOR TEST BED INTEGRATION ACTIVITIES

2.2.1 End-to-End Test Capability

The Space Station Information System (SSIS) comprises the information processing and communications capabilities that will be involved in handling operational and scientific data generated or used within the SSFP. The SSIS will provide end-to-end connection of, and a variety of information services to, a diverse and geographically distributed user community. This end-to-end connectivity will extend from the flight systems and payloads, to the ground-based support facilities, and on to university labs and international partner sites.

The role of the End-to-End Test Capability (ETC) is to support SSIS development efforts through advanced integration; that is, validation of operations concepts and demonstration of needed levels of interoperability in system designs (JSC, 1988^a). To this end, the ETC activities will require the integration of many SSFP test beds.

These efforts are complementary to, but do not overlap, the role of the Multi-System Integration Facility (MSIF). The MSIF performs integration testing on developed products prior to flight certification. The ETC is to provide an environment for early design activity, thereby clearing at least part of the path toward successful MSIF integration. The ETC is not concerned with flight qualification and it is not a formal verification environment (JSC, 1988^a).

Prior to ETC integration activities, most SSFP test beds functioned as stand-alone entities. The ETC attempts to integrate these test beds to test SSIS interoperability. An early focus of the ETC is to demonstrate prototype integrated operations of onboard systems. The goals of the ETC integration activities include the following objectives (Kelly, 1988^c):

- Development and testing of operations concepts for distributed, hierarchical management of Space Station Freedom systems, including OMS-system and ground-onboard functional partitions and interfaces
- Development of operations concepts for individual systems, including internal system functions and external interfaces
- Determination of requirements for real-time operations including communications, data handling, commanding, and caution and warning

The OMA node served as a focal point in early ETC demonstrations. Phase I consisted of the integration of the JSC DMS Test Bed, the OMA Node, and the GN&C Emulator Test Bed to perform a Space Station Freedom reboost scenario, leaving hooks and scars for adding other systems simulations as they become available. Figure 2-19 shows the ETC Phase I DMS/OMA/GN&C integrated testing schematic. During the execution of this scenario, if the Integrated Status Assessment prototype concludes that an action should be taken (i.e., a command should be sent), it sends a message to the Procedures Interpreter, which filters it, if necessary, and issues the appropriate commands using network communications services provided by the DMS Test Bed. These commands are sent to the GN&C Executive, hosted on a DEC MicroVAX on the GN&C Test Bed. The Procedures Interpreter may also request the transmission of data (e.g., status information) from the GN&C Test Bed. During procedure execution, the Procedures Interpreter monitors data from the systems, and by interpreting the status information provided, monitors the progress of the procedure (Kelly, 1988^a). The implementation of Phase I was completed and was widely demonstrated.

Phase I ETC integration exhibited the following accomplishments (Kelly, 1988^c):

- Autonomous system operations when appropriate
- Integration of Space Station Freedom activities by OMA when appropriate
- Different degrees of automation for OMA
- Different levels of caution and warning by OMA
- Crew override of OMA automation

Phase II ETC integration will functionally integrate additional system simulations and support nodes into the reboost scenario used in Phase I. Implementation of Phase II is in progress. Phase II integration, illustrated in Figure 2-20 by the shaded boxes, will include the following test beds and nodes (Ulmer, 1988, Kelly, 1988^c):

- JSC DMS Test Bed
- Phase II OMA Node
- GN&C Emulator Test Bed
- C&T Test Bed
- TCS Test Bed
- OMGA Node within the SSCC Test Bed
- GEPDC Test Bed
- HCI Test Bed
- D&C Test Bed

ETC Future Plans. Future ETC integration efforts will emphasize nodes that are remote to JSC and additional scenarios. The DMS Test Bed will become the equivalent of one node of a broader SSIS End-to-End Test Bed. Future integration and demonstrations could include the following (Kelly, 1988^c):

- Hierarchical OMS functions (e.g., via an *element OMS* capability at MSFC)
- Payload operations (e.g., processing telemetry from an onboard science instrument controlled by the University of Colorado at Boulder, or the integration of the Payload Operations Integration Center at MSFC)
- International partner participation
- Functions external to the manned base (e.g., the Platform Management System at GSFC)
- SSIS data management issues
- Additional OMS functions
- Ground/onboard interactions (OMA-OMGA)

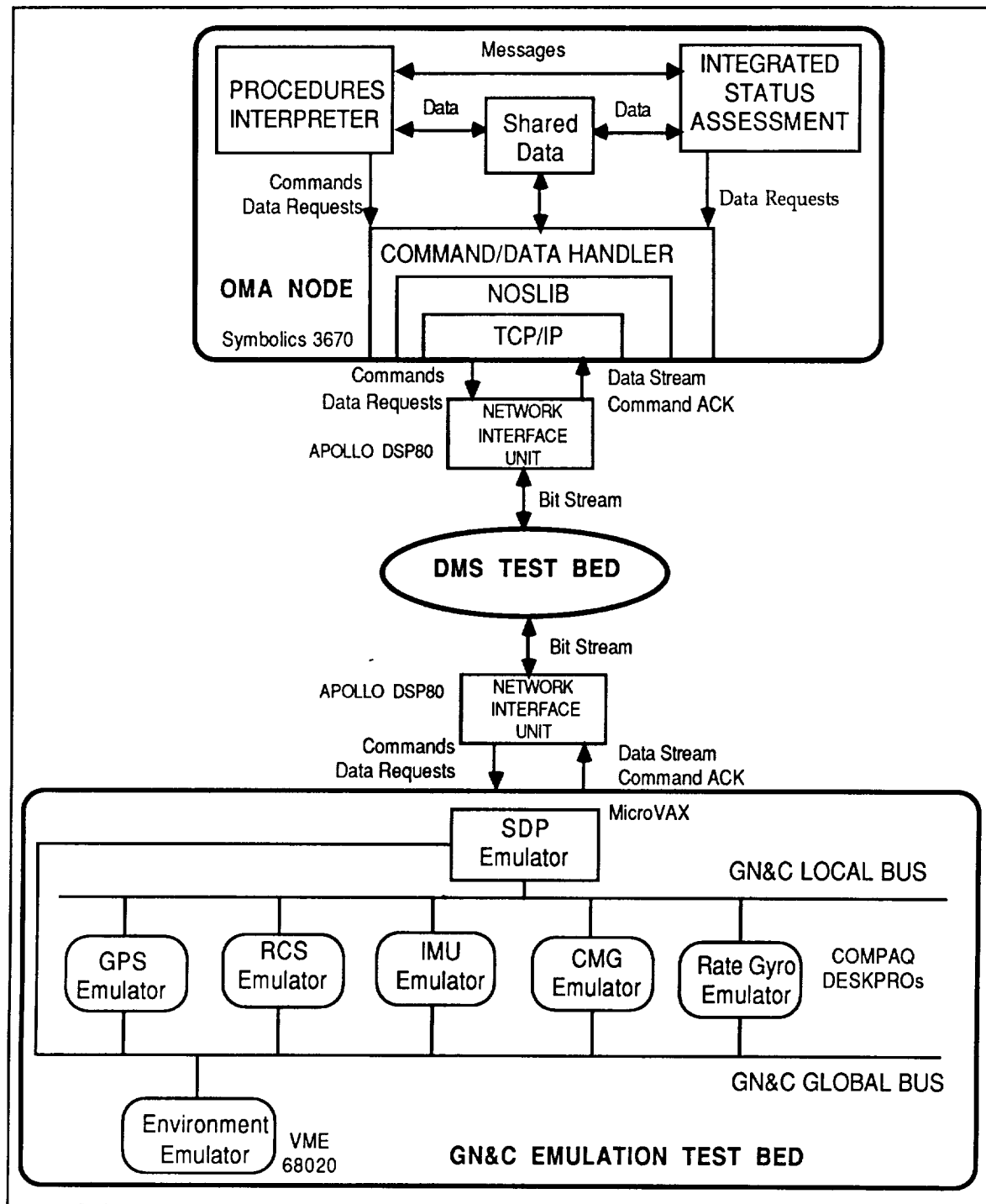


Figure 2-19. ETC Phase I Integration: DMS/OMA/GN&C
(Source: Kelly, 1988a)

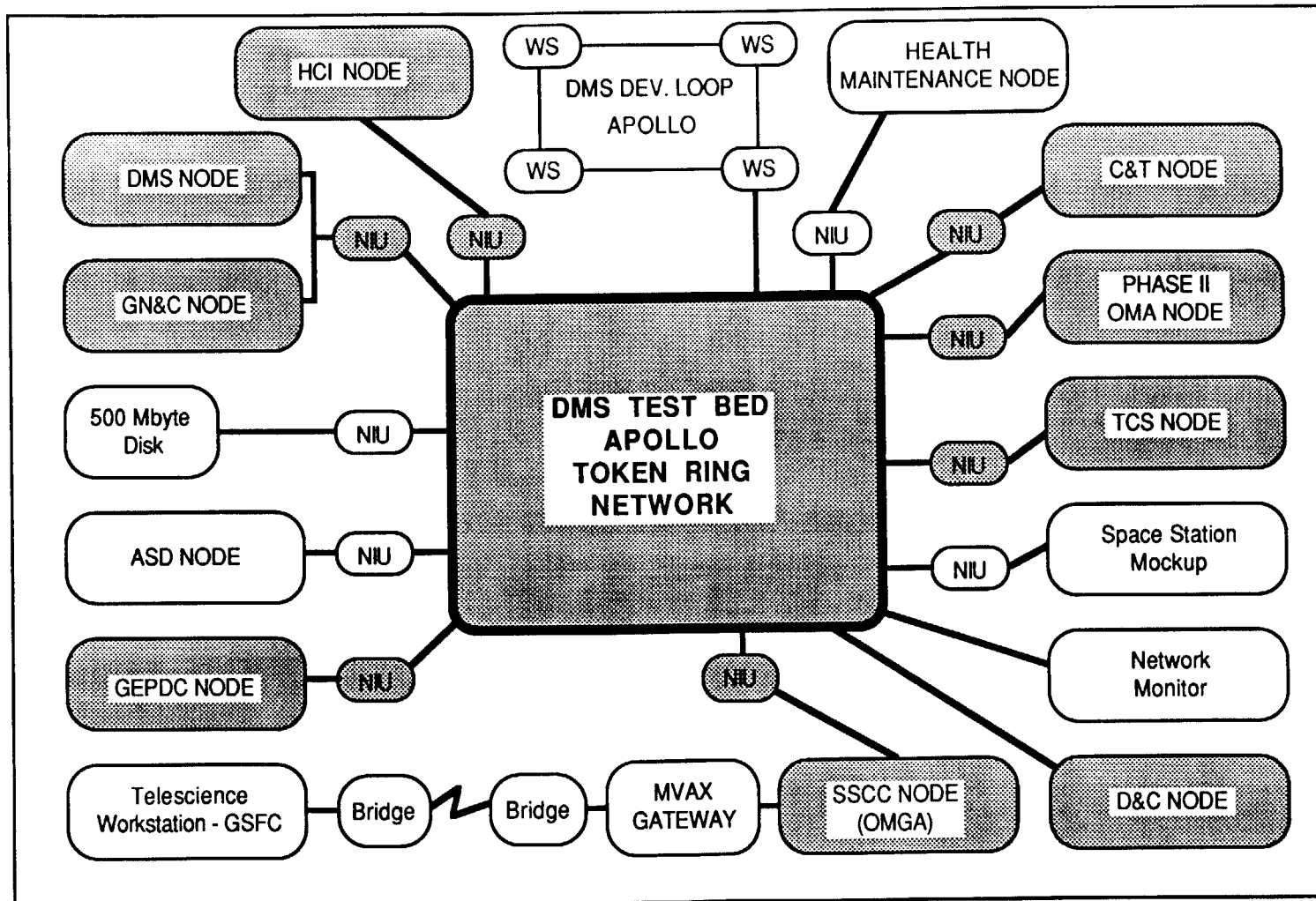


Figure 2-20. ETC Phase II Integration

2.2.2 Systems Autonomy Demonstration Project

The System Autonomy Demonstration Project (SADP) is funded by the NASA OAST as part of the SATP. This program's goal is to develop, integrate, and demonstrate the technology to enable intelligent autonomous systems for future NASA missions. The program objectives will be accomplished by a Core Technology research program closely coupled with several major demonstrations, the SADP. The critical core technologies include: task planning and reasoning, control execution, operator interface, and system architecture and integration. The core technology development will be implemented at NASA centers, universities, and industrial institutions. These efforts will be at the laboratory breadboard integration level and will be carried to the point where they can be transferred to technology demonstration projects. The SADP activities include a sequence of progressively more complex demonstrations. This sequence includes the intelligent control and operation of single subsystems in 1988, multiple subsystems in 1990, hierarchical multiple subsystems in 1993, and distributed multiple subsystems in 1996. A Memorandum of Understanding between OAST and OSS concerning advanced automation was signed in January 1988 with an accompanying Memorandum of Agreement which provides for the coordination and transfer of SADP-developed technology to the SSFP.

The 1988 single system demonstration will be a joint effort between ARC and JSC for the autonomous thermal control system operations for Space Station Freedom. The 1990 multiple system demonstration will be a joint effort between ARC, LeRC, MSFC, and JSC for the autonomous control of the thermal and electrical power systems for Space Station Freedom. The 1993 hierarchical demonstration and the 1996 distributed demonstration will evaluate and validate methodologies for expert system control of multiple subsystems through hierarchical and distributed architectural strategies, respectively (ARC, 1987^a; ARC, 1987^b).

2.2.2.1 1988 Single System Demonstration

The 1988 single system demonstration is a joint effort between JSC and ARC for autonomous TCS operations, and will use the Thermal Test Bed at JSC and associated capabilities at ARC. The major reason that the thermal system was chosen for this first demonstration was its slow dynamics. Controlling expert system software could be simpler than would be needed for a system with faster dynamics such as electrical power. According to the SATP Plan (ARC, 1987^a), the significance of this demonstration stems from its being the first NASA knowledge-based system to control a large complex system in real-time and with real operational software. The demonstrated software will provide advice on diagnosis and correction of faults, failure prevention through trends analysis, and explanation capabilities. Causal modeling, intelligent reasoning through causal models and heuristic rules, trend analysis, and validation methodologies are the key technology thrusts of this demonstration. Advanced automation software for this demonstration includes the TEXSYS developed jointly by JSC and ARC.

2.2.2.2 1990 Multi-System Demonstration

The multi-system demonstration will include the Thermal Test Bed at JSC, the EPS Test Bed at LeRC, the SSM/PMAD Test Bed at MSFC, and associated capabilities at ARC. The 1990 activities will actually be a series of demonstrations, each building on the accomplishments of earlier demonstrations. The capabilities demonstrated will be the autonomous control of both the thermal and power systems' operations. Technologies to be demonstrated include fault detection, classification and isolation methodologies, system restoration strategies, replanning under uncertainty, and operator training methodologies. Expert system software will control the operations of the participating systems and will demonstrate the technologies listed above (Wong et al., 1988).

The first demonstration will involve the electrical power system. The LeRC EPS Test Bed will serve as the power generator of the EPS; the system will be controlled by the PMACS expert system. In the second demonstration, the MSFC SSM/PMAD test bed will act as a *smart load* on the LeRC power source, simulating the power distribution requirements of a Space Station Freedom module, such as a habitation module. In addition to the load of the MSFC Test Bed, LeRC will have the power loads of attached payloads. Expert system software at LeRC (PMACS) and MSFC will control the operations of these power systems.

The final demonstration will bring together the Thermal Test Bed at JSC, the two power test beds at LeRC and MSFC, and capabilities at ARC as shown in Figure 2-21. ARC is responsible for directing project management expert system development. All participating systems will be connected via network links.

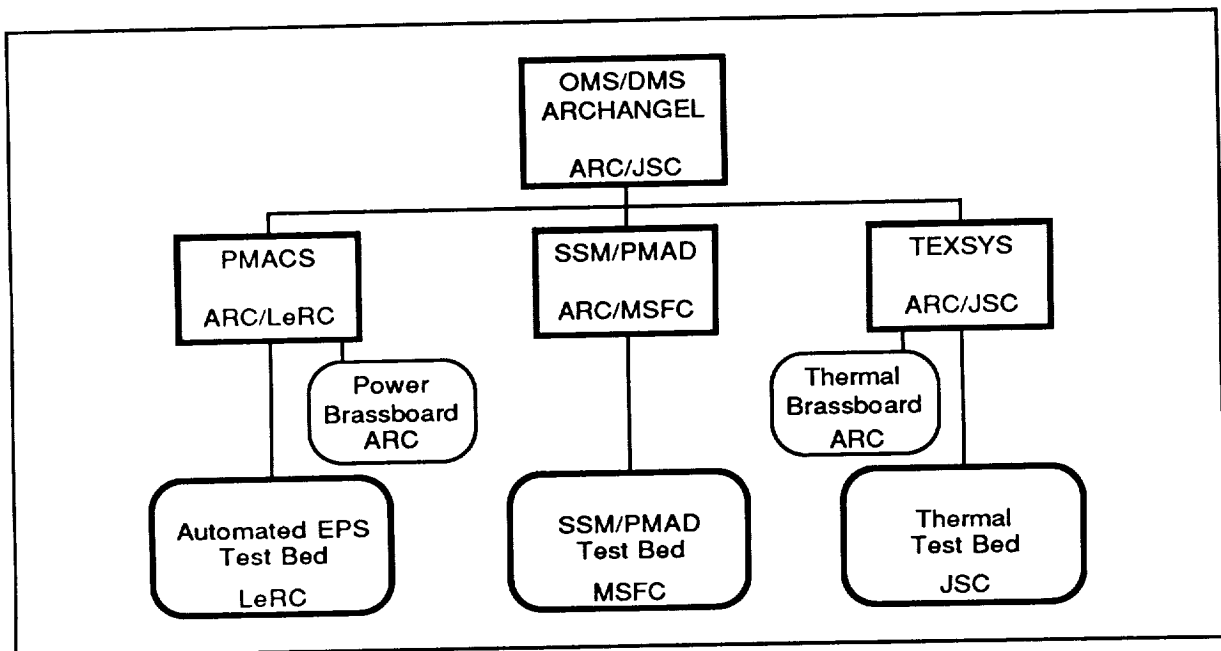


Figure 2-21. 1990 SADP Demonstration: Cooperating Systems
(Source: Wong et al., 1988)

The cooperative interaction of these systems will be through an intelligent executive controller called the Autonomous Real-time Control of Hierarchical Agents and Networks at the Global Executive Level (ARCHANGEL). ARCHANGEL will be developed at ARC and hosted on the DMS Test Bed at JSC. It will be the functional equivalent of the OMS and has the potential to influence the OMS architecture.

2.2.2.3 1993 Hierarchical System Demonstration

This demonstration will evaluate and validate methodologies for expert systems control of more than two Space Station Freedom subsystems through hierarchical architectural strategies.

2.2.2.4 1996 Distributed System Demonstration

The purpose of the 1996 demonstration is to evaluate and validate methodologies for expert systems control of multiple Space Station Freedom subsystems through distributed architectural strategies.

2.2.2.5 Issues

The most important issue raised in our interviews was cooperative problem solving. If these test bed integration efforts are to be technically successful, effective and open communication and coordination are essential. This is a technical issue relative to technology transfer, cooperative problem solving, and coordination for commonality, but it is equally a management issue with respect to the establishment of effective means of communication and cooperation.

ARC is expecting the delivery of a miniature version of the LeRC EPS breadboard, similar to the one they received from JSC of the thermal system. LeRC's plans are somewhat different; they would like to provide high-fidelity models. The power system breadboard components are expensive. LeRC personnel believe that models are natural product of Space Station Freedom design efforts and are sufficient for systems design development, useful for verification, and provide an acceptable risk until test beds get closer to real space hardware.

2.2.3 Space Station Control Center Test Bed

The Space Station Control Center (SSCC), an SSFP ground facility at JSC, will be responsible for tactical and execution level planning and integration of manned base and user systems operations, monitoring, control, and configuration management of the Space Station Freedom core systems, storage and retrieval of core systems data, the overall integrity of the manned base, and the safety of the flight crew (JSC, 1988^d).

The Systems Development Division (SDD) within the Mission Support Directorate (MSD) and the Mission Operations Directorate (MOD) at JSC are jointly responsible for the definition, development, and implementation of the SSCC. MSD is building a SSCC Test Bed to provide an environment into which project specific prototypes and breadboards, and COTS products, can be introduced for evaluation. In August, 1988, an SSCC Test Bed Planning Requirements Document was distributed for review and comments. Because the existing SSCC Test Bed is very limited in its capability to support all the prototyping and evaluation activity that is required by the various disciplines within SDD, additional requirements were solicited relative to test bed enhancements (JSC, 1988^e).

The JSC SSCC Test Bed has connectivity to the overall SSIS ETC test beds and has gateways to external networks such as the Goddard Telescience Test Bed. Many SSCC critical design issues can be evaluated in a stand-alone environment; however, the structure of the integrated ETC environment will be utilized to support testing of several levels of SSCC critical design issues and concepts within the following areas (JSC, 1988^e):

- Common human-computer interfaces
- Network communications
- SSCC network management
- OMA/OMGA partitioning
- Transaction management
- Process-to-process communications
- Ground data distribution services
- Commonality of display generation and distribution
- Data base management services and architectures

- Security and privacy

Physical Test Bed. The hardware elements of the SSCC Test Bed, as well as physical connections to existing JSC test beds, are shown in Figure 2-22. The physical test bed currently includes the following computing hardware:

- Four Sun Microsystems workstations
- DEC MicroVAX II computer (gateway)
- DEC VAX 11/785 computer
- Apollo DSP-80 computer (NIU)
- Bridge Communications GS/3 commercial gateway server

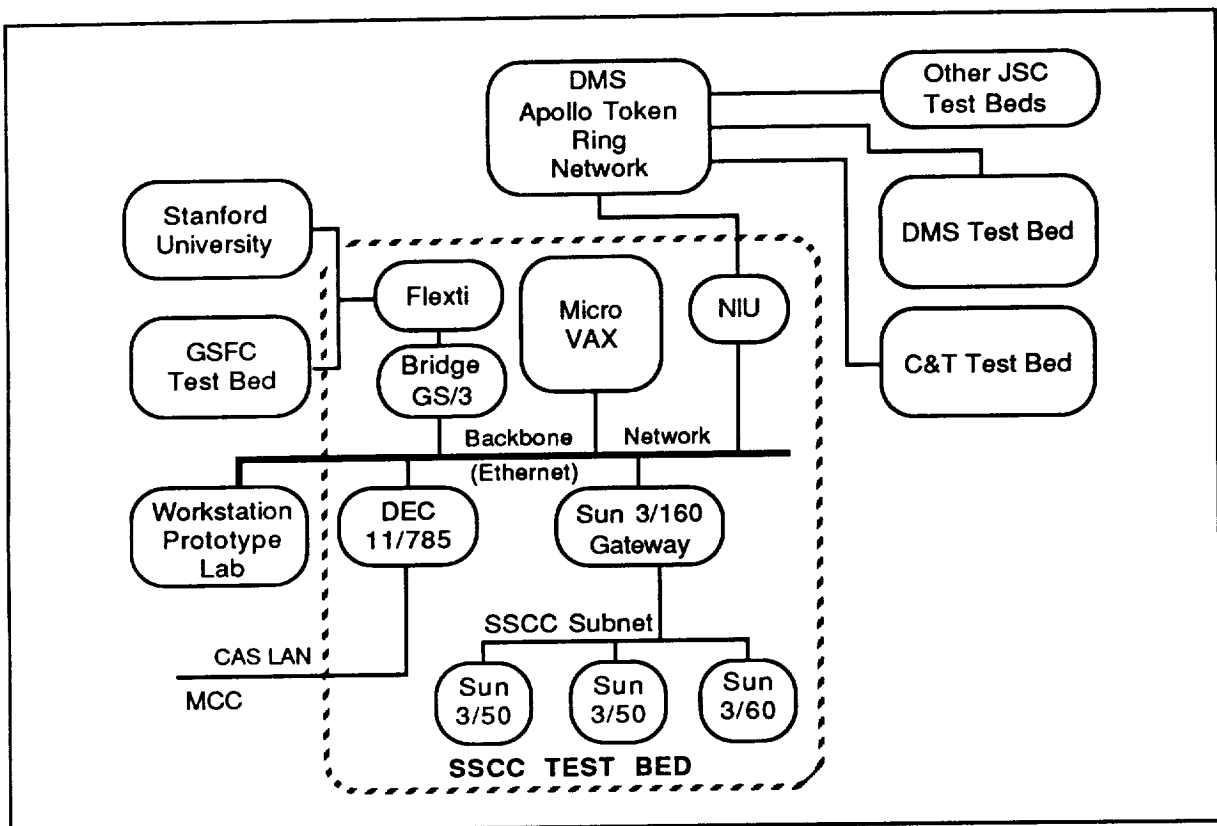


Figure 2-22. SSCC Test Bed Architecture
(Source: JSC, 1988^e)

The Apollo DSP-80 serves as a NIU to the Apollo token ring network. The SSCC Test Bed network contains two Ethernet LANs providing connectivity to all of the SSCC computers. The primary SSCC Test Bed Ethernet LAN connects the SSCC Test Bed with the SDD Workstation Prototype Lab (WPL). Two SSCC network gateways provide connections to the Program Support Communications Network (PSCN) and to the DMS Test Bed network. The secondary SSCC LAN provides connections only for the four Sun workstations within the SSCC Test Bed (JSC, 1988^e).

The existing SSCC Test Bed is expected to evolve in functionality to that of the operational SSCC. Upgrade plans include the replacement of the Ethernet LAN with an FDDI LAN to be compatible with the planned upgrades of the DMS Test Bed.

Logical Environment. The SSCC Test Bed and its Ethernet network backbone provide the logical functions of the operational SSCC. The GSFC and the Stanford test beds serve as remote site science and telecommunications stations. Logically, the DMS Apollo Domain Network represents the SSFP onboard DMS network and the DMS Test Bed provides the logical functions of the onboard DMS. The C&T Test Bed logically provides the control and monitoring capabilities of the onboard C&T. The Payload Simulator (PLS) Test Bed serves as the payload data source (JSC, 1988^e).

Software. The SSCC Test Bed provides a gateway program (NOSGATE) that translates between NOS and TCP/IP protocols. The test bed software also includes a COTS desktop publishing package by Interleaf Corporation, a NASA-built display builder and editor package (Prototype Display Builder), and environments for software development supporting the C, FORTRAN, Pascal, and Verdex Ada languages. The X-Windows window manager is also supported on the Sun workstations (JSC, 1988^e).

Advanced Automation. An OMGA prototype is being implemented on a Sun workstation. The OMGA will duplicate some of the OMA functions, allow control of the Space Station Freedom from the ground, and provide additional services such as generation of the Short Term Plan. The OMGA prototype consists of display and control prototype software for ground operations of the Space Station Freedom (Ulmer, 1988).

Existing Capabilities. Initial SSCC Test Bed efforts were focused on connectivity and interoperability to achieve a distributed test bed environment. This connectivity included the SSCC Test Bed, the DMS Test Bed, the C&T Test Bed and the PLS Test Bed. Data links to GSFC, WPL, and to Stanford University were also established. Successful prototypes were developed within these test beds that bridge communications between dissimilar protocols to establish communication transparency (JSC, 1988^e).

Planned Capabilities. Additional planned physical and functional capabilities include the following (JSC, 1988^e)

- Connectivity to the Software Support Environment (SSE) for access to development tools and environments
- Network communication upgrades for compatibility with the DMS network and added functionality
- Installation of an Radio Frequency (RF) link emulator
- Functional capability to collect and distribute telemetry data structures within the SSCC network
- Data base machine and relational data base management system
- Ada runtime executive for integrated process communication between applications

2.3 OTHER SIGNIFICANT RESEARCH ACTIVITIES

2.3.1 INCO Expert System Project

One part of the Systems Autonomy Technology Program consists of specific domain demonstrations. A set of these demonstrations has been planned to facilitate technology transfer to domains other than the Space Station Freedom. One of these demonstrations is the INCO Expert System Support Project (IESP). This demonstration is significant in that it will be the first NASA knowledge-based system to be implemented into a real-time operational environment (ARC, 1987^a). IESP is managed by the MOD at JSC and is being developed by an MOD contractor team (including The MITRE Corporation).

MOD has as one of its responsibilities the operation of the Mission Control Center (MCC) at JSC in support of Space Shuttle ground operations. MOD will have a similar role in the operations of the SSCC and the Orbital Maneuvering Vehicle (OMV) Control Center. MOD is investigating and evaluating methods of attaining increased automation in support of flight controllers.

The objective of IESP is to provide assistance to the INCOs in the management (i.e., control and monitoring) of two-way communication between the ground and the Space Shuttle in orbit, as well as between the Space Shuttle and its payloads. An INCO fills a *front-room* position within the MCC. On a normal Space Shuttle mission there are usually one or two INCO's in the front room and 2 or 3 in the *back room* of the MCC. The INCOs' job is to monitor console displays which present real-time information about the communication links (voice, video, digital data) between the ground and the shuttle, and to send commands to communication devices to keep the links working properly. These communication links comprise three different data paths: S-band (no video), KU-band (high frequency, high data-rate), and UHF (voice only).

Currently, the displays used are simple monochrome CRTs with cluttered character output. The values listed on the screen are related to the status of various communication links and devices. One goal of this project was to replace this cluttered, character-only display with a more easily interpreted, interactive, point-and-click type of display which would present more useful, easier-to-identify information to the INCO using color and graphics, to allow the INCO to more easily control the communications devices. A second goal was to provide expert analysis, interpretation, and explanation functions to assist the INCO in assessing communication link malfunctions and configuration problems.

The INCO expert system attempts to emulate the responses of INCOs to communication stream malfunctions and configuration problems. It works within a fairly realistic temporal constraint of 15 seconds for common solution formulation, a reasonable time period for a human INCO with 6 months to 1 year of training on a particular payload. The expert system has been installed within the MCC and resides next to the INCOs' workstations. It will be evaluated by the INCOs, a test of its performance and acceptance, during the flight of STS-26, of the Space Shuttle.

The INCO expert system is a working prototype; however, it will be continually modified and extended. It is written in CLIPS, an expert system shell developed by the Mission Planning and Analysis Division of the Mission Support Directorate at JSC. Parts of the system were developed using the high-level language C. IESP was developed on a Motorola 68020-based Masscomp computer with 12 megabytes of RAM, running the UNIX operating system.

Current and planned activities include the development of a Telemetry Data Base Verification expert system and upgrades to existing equipment, including display equipment within the MCC and computing equipment used to develop and run the IESP software.

The INCO expert system runs in real time in the sense that it analyzes real-time data and formulates answers in real time. The Telemetry Data Base Verification expert system will be used prior to NSTS

missions to verify the Shuttle telemetry formats which will be used on an upcoming mission. Up to 20 different data formats, from hundreds of possible combinations, are used during various phases of a Shuttle mission; these formats are defined and stored on a Shuttle data tape. This tape is converted to a data base for use by IESP. It is important that these formats be verified in the sense that IESP must conform to these formats and mission phase-related format changes. This expert system project is also being developed using CLIPS and C on a Masscomp computer.

Equipment upgrades include the replacement of MCC displays with color monitors which support the output of the IESP applications and a larger telemetry system which will increase the number of telemetry parameters which can be captured from 4000 to 15000 parameters. These upgrades, along with Ethernet networking, will allow multiple, distributed nodes and larger numbers of both users and telemetry parameters which can be selectively distributed to those users. At least one more Masscomp computer will be purchased to support these upgrades.

The data gathering capability of the IESP will be used also by ground controllers responsible for the Shuttle main engines. IESP telemetry software, along with engine monitoring and control software, will help these controllers monitor main engine problems when they arise and make decisions concerning corrective actions such as shutting off a main engine, if necessary.

2.3.2 Transition Flight Control Room

The Transition Flight Control Room (TFCR) at JSC is located adjacent to the MCC and is an engineering test bed for control room hardware and software systems in a near operational environment. A major function of the TFCR is support for MCC Upgrade (MCCU), the augmentation and replacement of major MCC hardware and software. Although the TFCR is closely related to the MCC, it is not solely an MCC facility. The TFCR serves as a generalized control center test bed environment. It provides a demonstration facility for design approaches, allows the validation of user operational requirements, and the transition of new technologies into flight control rooms. The TFCR allows flight controllers to evaluate proposed upgrades under near operational conditions (e.g., using real telemetry data). Real-time telemetry is available to workstations in the TFCR through a 100 MBS backbone LAN; a general purpose LAN connects the workstations in the TFCR.

Equipment being tested in the TFCR includes intelligent workstations, color graphics displays, physical input devices (e.g., touch screens, button panels), and front screen projectors for display of workstation generated data, graphics, and video. Software includes windowing environments (e.g., the Windowing Executive -- WEX), a Data Display Executive that provides connectivity between data sources and data displays), several display builders/managers, and an Interim Data Manager that provides acquisition of real-time data by application software.

An Integrated Workstation Prototype project will package MCCU-specific and commercial technologies in an integrated environment. These activities include a Projection Plotting Display (PPD) Application (including a wide screen display for world map, ascent/entry, and artwork), TV window overlay on PPD output, an ADS (on-line flight manuals and procedures stored on laser disks, user report and documentation development, optical character reading, and monochrome graphics scanner), PC emulation (PC-DOS), color image scanner and display, and color printer with screen capture.

The TFCR is actively used during Space Shuttle missions. Because of the periodic nature of the NSTS, the TFCR could be applied to SSFP activities between Shuttle missions. A TFCR interface to SSFP data is being defined.

SECTION 3

OPERATIONAL AND SUPPORT CAPABILITIES

The following sections discuss the operational and support capabilities of the Space Station Freedom Program (SSFP) with respect to formal software development, test, and integration. In Section 3.1 the Software Support Environment, the designated environment for Space Station Freedom software development, is discussed. Although the reader is expected to have some background on the role and purpose of the SSE, Section 3.1.1 provides a brief overview of the SSE and its elements. Each of these elements is then discussed in more detail. Section 3.2 introduces the Multi-System Integration Facility

3.1 SOFTWARE SUPPORT ENVIRONMENT

Space Station Freedom software will be developed by a cast of thousands, spread across multiple NASA centers, contractor facilities, and international partners' facilities. To provide as much uniformity and consistency as possible for the software development at NASA and contractor facilities, an overall framework called the Software Support Environment (SSE) has been defined. Applications developed by international partners will be developed outside of an SSE-supported environment.

The SSE is not a single computer facility or even a set of homogeneous facilities. As the name suggests, it is an environment and a set of policies, procedures, and tools that can be installed on the tangible Software Production Facilities distributed throughout the SSFP community. The objectives of the SSE are to provide a single logical environment (with multiple physical instantiations) with the following capabilities:

- Support for all Space Station Freedom operational software
- Full functionality to support software from conception to retirement
- Economical maintenance
- Technological currency during its lifetime

The SSE physical entities consist of a single Software Support Environment Development Facility (SSEDF), located at JSC, at which the Space Station Freedom software development policy and tools (*rules and tools*) are baselined, and multiple facilities (Software Production Facilities--SPFs) on which these tools are installed for facilitating Space Station Freedom software production (see Figure 3-1).

The SSE contractor (Lockheed Missiles and Space Company of the Lockheed Electronics Corporation) is responsible for implementing the SSE architecture that is shown in Figure 3-2. The conceptual architecture shown in this figure depicts a *Process Management Element* that isolates the toolset from the supporting host system. This element will be unique to each vendor-specific instantiation of the SSE; thus one SPF could host SSE tools on an IBM mainframe, while another SPF could host the same toolset on a DEC VAX. In addition to ensuring portability among platforms, the Process Management element is intended to provide for eventual inclusion of methodologies and technologies such as structured design methodologies, object-oriented programming techniques, and artificial intelligence. The *Software Production and Integration, Test and Verification Elements* are of primary concern in the evaluation of the SSE for advanced automation software development.

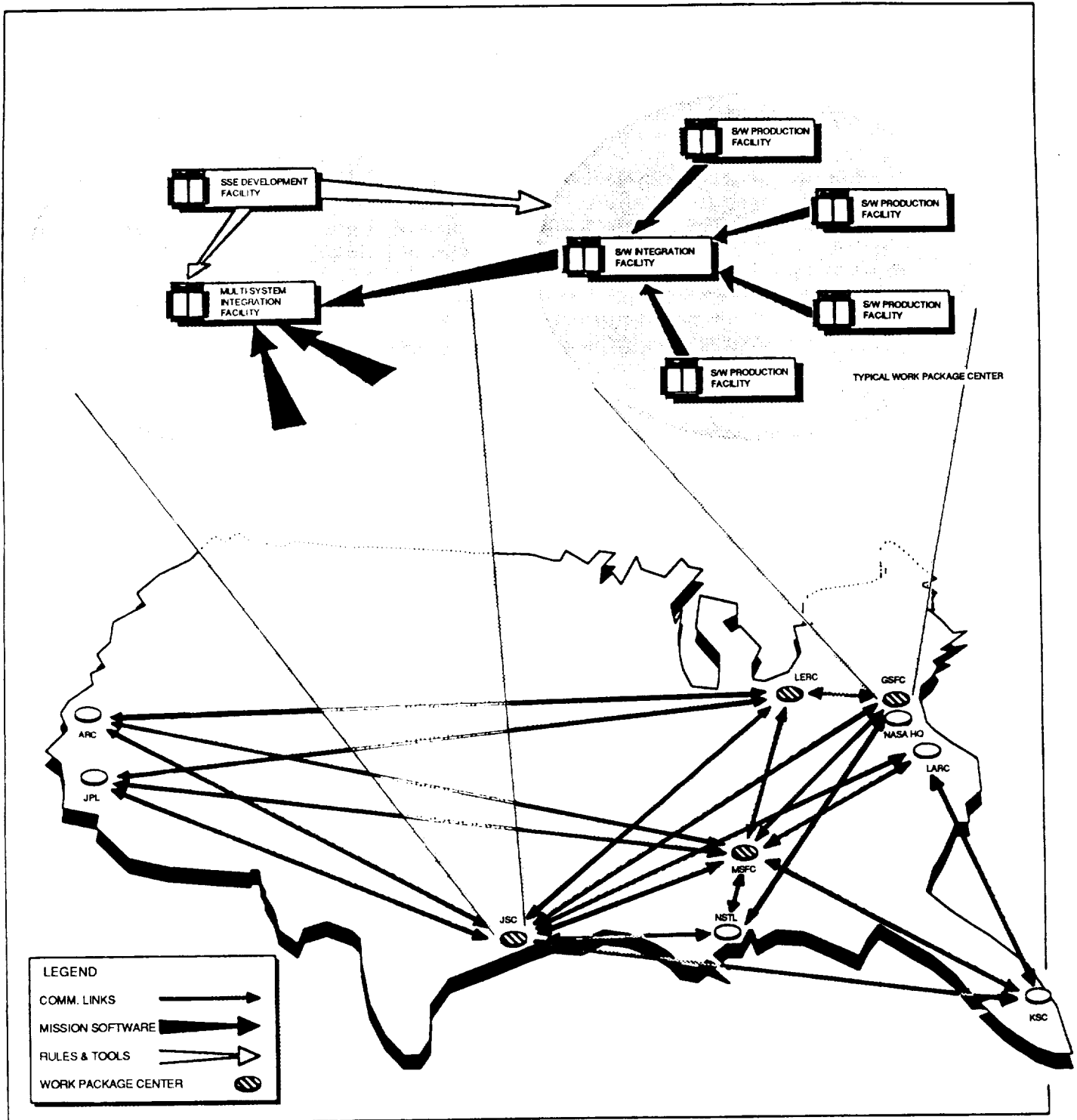


Figure 3-1. Operational and Support Facilities
(Source: SSP, July 1988g)

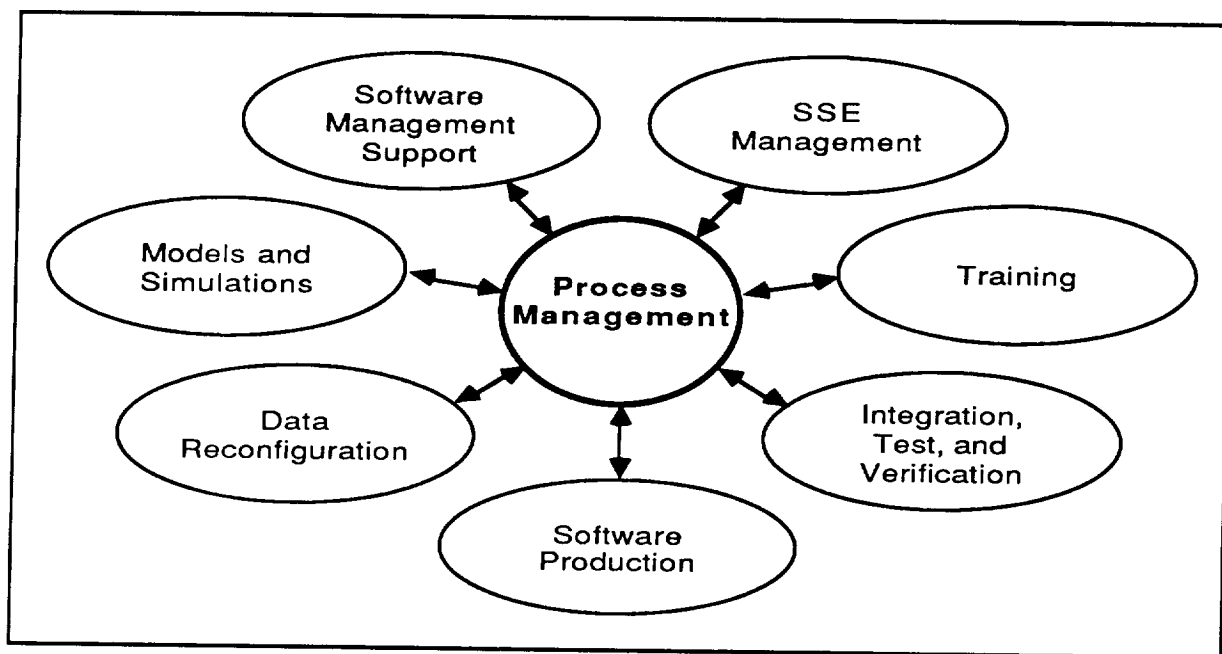


Figure 3-2. SSE Reference Architecture
(Source: LMSC, 1988^a)

3.1.1 Software Support Environment Development Facility

3.1.1.1 General Description.

The SSEDf develops the rules and tools that are exported to the SPFs. In order to avoid extensive new development, the SSEDf is expected to rely mostly on Commercial-off-the-shelf (COTS) software (Computer-Aided Software Engineering tools, project management tools, software development tools, etc.) to satisfy the toolset requirements. Custom-built software necessary to satisfy unique processing or control requirements will be written in the Ada programming language. The SSEDf toolset will support the various SPFs in all phases of the software life cycle; use of the SSEDf tools will require that individual SPFs consist of hardware and operating system software that can host the tool set.

In addition to providing a toolset, the SSEDf will be a single repository of common models of Space Station Freedom systems (perhaps initially in the form of functional simulations) which will be developed at individual work package SPFs and then sent to the SSEDf for configuration management. These models will be used for early test and verification purposes, and will be contained in the *Models and Simulation Element* (see Figure 3-2).

3.1.1.2 Advanced Automation Capabilities.

The baseline requirements for SSE advanced automation tools are contained in (LMSC, 1988^b; LMSC, 1988^a). These documents set forth functional (and in some cases detailed) requirements for the various SSE elements; the expert system tool requirements are set forth in the sections describing the Software Production Element. Briefly, they require that the SSE provide the following capabilities:

- Develop and deliver expert system applications

- Provide incremental compilation of rules
- Provide expert system generation capabilities
- Provide interfaces between the expert systems and other SSE applications
- Provide for the generation of expert systems that run using SSFP flight software or Space Station Information System (SSIS) production ground elements

These requirements, as they currently stand, could possibly be satisfied by one or more COTS knowledge-based shells. Several off-the-shelf tools should be available by the time the SSEDf reaches its first major Operational Increment milestone (tentatively December 1989). This milestone will provide the first major toolset *load* to the SPFs, including an expert system tool. If satisfaction of the COTS requirement is not possible, and the requirements mandate a development effort, one would imagine such tools would have to be Ada-based (based on stated SSE policy). The need for such Ada-based development becomes increasingly unlikely as time passes and more knowledge-based shells for standard computers are marketed.

3.1.2 Software Production Facilities

3.1.2.1 General Description

Space Station Freedom development is relegated to the various NASA Centers by way of Work Packages (WP) (i.e., WP1 is managed by Marshall Space Flight Center -- MSFC, WP2 by Johnson Space Center -- JSC, WP3 by Goddard Space Flight Center -- GSFC, and WP4 by Lewis Research Center -- LeRC). The embedded software for the Space Station Freedom systems and elements that the WPs are responsible for will be produced on the various WP SPFs. The support software configuration of these SPFs and the rules and tools under which application software is produced will be provided by the SSEDf. The SPFs, while physically separate from the SSEDf, will provide a common host environment for the SSE tools across the SSFP for software development, test, and maintenance. The SPFs will utilize a variety of computer equipment in discharging their flight software development responsibilities. The current baseline equipment includes mainframes (IBM-compatible hosts, DEC VAXs, and possibly other hosts using the UNIX operating system) and workstations (Macintosh IIs, IBM PC compatibles, and Apollos). Individual SPFs will develop software specific to their work package responsibility, using models (functional simulations) of other work package systems as needed for integrated testing. For example, LeRC will perform integrated testing of power system software using software services and flight-like hardware provided by the JSC Data Management System (DMS).

To perform this integration and testing role, the SPFs will be enhanced with *DMS Kits*; this enhancement will allow the SPF to play the role of a System Integration Facility (SIF) for comprehensive intra-work package test and verification purposes. Briefly stated, "a DMS kit will contain the hardware and DMS support software contained and utilized by the distributed system (or elements, i.e., the flight article), as well as the hardware and software needed for the DMS support environment" (JSC, 1987).

3.1.2.2 Advanced Automation Capabilities

The design philosophy of the SSE is that the SSEDf toolset should migrate to new computer architectures as they enter the mainstream. Thus, while the early SPFs will be IBM-compatible or DEC VAX hosts, the goal is to accommodate new host architectures as they become proven. Probably, the first such SPF re-host will be to a UNIX host. Further down the road, one can imagine SPFs with parallel architectures (e.g., the Sequent computer line) hosting the SSEDf toolset.

Similar projections can be made for the SPF workstations. While the SSE workstation platforms (Macintosh II, Apollo, IBM PC) as currently configured are not suitable for stand-alone advanced au-

tomation development, they could easily be reconfigured to change that situation by including hardware extensions (e.g., Lisp boards and more memory) and software tools that satisfy the SSEDf expert system requirements. This reconfiguration would have to be done in the context of preserving the *baselined* workstation configuration as an operable subset. The availability of *plug-in* AI development environments (e.g., TI Explorer board on the Macintosh II and the Gold Hill Hummingboard and Gold Hill development system on the IBM PC) to be added to conventional workstations makes this possible. Thus, a rather complete Artificial Intelligence (AI) development environment (almost equivalent to a stand-alone Symbolics machine) could be configured on an existing SSE workstation.

As expert system development environments migrate to mainframes, similar extensions should be available for the SPF hosts. Intellicorp already provides a version of KEE for the IBM mainframe and a wide variety of expert system tools run on DEC VAXs.

3.1.3 Verification and Validation Capabilities

The following paragraphs briefly discuss the approach to SSFP verification and validation (V&V) starting with a general description of the proposed SSFP V&V process, followed by a more focused discussion of how SSFP testing will be accomplished, and concluding with the current testing architecture and V&V requirements relative to advanced automation support.

3.1.3.1 Verification and Validation Process

SSFP elements, systems and software will be tested, integrated, and verified by means of a V&V methodology that spans the SSFP, from the Level II Program Office to the contractors and suppliers. In this hierarchical process, Level II is responsible for developing the top level SSFP V&V requirements and plans, for verifying the integrated Space Station Freedom software, and for producing the flight software load. Level III is responsible for developing derived V&V requirements, for developing and implementing V&V plans, and for verifying distributed systems and elements. Finally, the contractor or supplier is responsible for the performance of all Orbital Replaceable Unit (ORU) and subsystem hardware and software V&V. The policy governing V&V is covered in (SSP, 1988e; SSP,

The V&V process encompasses three phases. The first phase, *development*, ensures that the proposed hardware and software design concepts are acceptable for use in the intended application, and that technical, cost, and schedule risks which may be encountered during V&V are minimized. The *certification* phase of the V&V process is meant to ensure that the hardware and software complies with all design and operational requirements, including all design and safety margins. Finally, the *acceptance* phase determines that the hardware and software is built to its specifications.

The three phases of V&V are applicable to each of the test levels (ORU through onboard V&V) shown in Figure 3-3, but the V&V methods, equipment, and location may change. To illustrate differences in V&V methods, development phase V&V at the system level may use a system test bed; this would be an inappropriate method for multi-system integration. To illustrate differences in V&V equipment, the need for, or nature of, a DMS kit would depend on the level of testing.

3.1.3.2 Testing Approach

Several V&V methods are acceptable for Space Station Freedom use, governed by criticality (personnel and SSFP element critical, mission critical, and all others). These are demonstration, inspection, analysis, and test (or combinations of these four). A V&V process flow relative to

ITV LEVEL	TEST LEVELS	SCOPE			PURPOSE OF TEST
		HARDWARE	SOFTWARE	MODELS/SIMS	
0	DEVELOPMENT	<ul style="list-style-type: none"> CARDS BREADBOARDS PROTOTYPES 	<ul style="list-style-type: none"> UNITS MODULES 	PREL MODELS & SIMS	<ul style="list-style-type: none"> VALIDATE DESIGN CONCEPTS PROOF OF DESIGN
1	ORU	<ul style="list-style-type: none"> ORU MECHANISM 	<ul style="list-style-type: none"> ORU LEVEL APPL. SW 	ORU MODELS	<ul style="list-style-type: none"> ACCEPTANCE QUAL BIT EFFECTIVENESS
2	ASSEMBLY	<ul style="list-style-type: none"> SEVERAL ORUS PALLET 	<ul style="list-style-type: none"> MULTI-ORU APPL. SW 	MODEL OF ASSEMBLY OR PALLET	<ul style="list-style-type: none"> ACCEPTANCE REDUNDANCY MGT. VERIF.
3	SUBSYSTEM	MAJOR COMPLEMENT OF ORUS	<ul style="list-style-type: none"> SUBSYSTEM APPL. SW 	SUBSYSTEM MODELS & SIMULATIONS (GROUND & FLIGHT)	<ul style="list-style-type: none"> DESIGN ACCEPTANCE QUAL
4	SYSTEM	PRIMARY GROUPING <ul style="list-style-type: none"> VERTICAL SYSTEM TESTS 	<ul style="list-style-type: none"> SYSTEM APPL. SW 	TOTAL SYSTEM MODEL/SIMS	<ul style="list-style-type: none"> DESIGN QUAL DD-250 SELL OFF
5	MULTI-SYSTEM OR ELEMENT	<ul style="list-style-type: none"> MULTI-SYSTEMS MSIF IACO 	<ul style="list-style-type: none"> MULTI-SYSTEM APPL. SW END-END TESTS 	ELEMENT OR MULTI-SYSTEM MODELS & SIMS	<ul style="list-style-type: none"> HORIZONTAL SYSTEM TESTS/VERIFICATION ELEMENT C/O
6	LAUNCH READINESS	<ul style="list-style-type: none"> HW IN LAUNCH PACKAGE 	<ul style="list-style-type: none"> SUBSET OF SW IN LAUNCH PACKAGE 	FLIGHT SIMS	<ul style="list-style-type: none"> PRE-FLT. C/O
7	ON-ORBIT (TO IOC)	<ul style="list-style-type: none"> ON-ORBIT C/O & VERIFICATION 	<ul style="list-style-type: none"> ORBITAL SW CONFIGURATION 	MODELS/SIMS OF FLT. CONFIGURATION	<ul style="list-style-type: none"> SUPPORT ON-ORBIT BUILD UP & C/O
8	ON-ORBIT LONG TERM OPS. / C/O	<ul style="list-style-type: none"> ORBIT/GROUND HW CONFIGURATION 	<ul style="list-style-type: none"> ORBIT/GROUND SW CONFIGURATION 	ORBIT/GROUND HW/SW MODELS/SIMS	<ul style="list-style-type: none"> SUPPORT ON-GOING ORBITAL FLIGHT

Figure 3-3. Integration, Test and Verification Levels
(Source: JSC-32795, 1987)

each of these is set forth in (MDC, 1988^b). Independent verification and validation (IV&V) activities (in addition to V&V activities) are specified for the highest criticality software products. V&V activities in the SSFP include:

- Analysis of requirements
- Analysis of code (through inspection or automated analysis aides such as static and dynamic analysis tools)
- Evaluation of documentation, system performance, and hardware and software compatibility
- Provision of procedures and plans for V&V tests
- Execution of those tests

The SSE provides tools for accomplishing the above activities as part of its Integration, Test and Verification Element (LMSC, 1988^a). These tools will be made available as part of the uniform delivery of toolset capabilities from the SSEDF to the SPFs. These tools include the following capabilities:

- Pre-test setup tools
- Models of systems
- Execution tools
- Post-test analysis tools
- Test library setup functions
- Integration capabilities

Each SSFP WP participant is responsible for establishing a Verification Data Base (VDB) in a common SSFP Office Verification Office format to provide program management visibility into the overall SSFP verification effort.

3.1.3.3 Testing Architecture

Because of the distributed nature of the SSFP, a test and integration architecture presented a special problem area. JSC recognized these problems during the Phase B time frame, and suggested a novel approach for software development, testing and integration called Test And Verification of Remote Networked Systems (TAVERNNS). At the heart of TAVERNNS is a framework that provides the various SPFs (each concerned with its own systems) with flight host hardware and with portable models of the rest of the Space Station Freedom systems. This environment consists of three parts as shown in Figure 3-4: a Test Control and Simulation Environment (TCAS), the System Under Test (SUT), and the DMS Environment (JSC, 1987).

The TCAS environment is provided by the SSE Independent Test & Verification (IT&V) Element. The SUT environment and the DMS environment are coupled by the use of DMS kits. Relative to the SUT, the DMS kit provides hardware and software services to the system being implemented; the same hardware and software services that this system will be hosted on and require while on orbit (actually, the hardware will not be flight qualified, but will otherwise be equivalent). Relative to the DMS environment, DMS kits provide an interface to the SSE models (see Figure 3-5).

DMS kits will be distributed to all Space Station Freedom Software developers and will provide, along with common simulation models, a test environment in which DMS services and *missing* systems can be included in the test process at the Work Package Software Production Facility/Software Integration

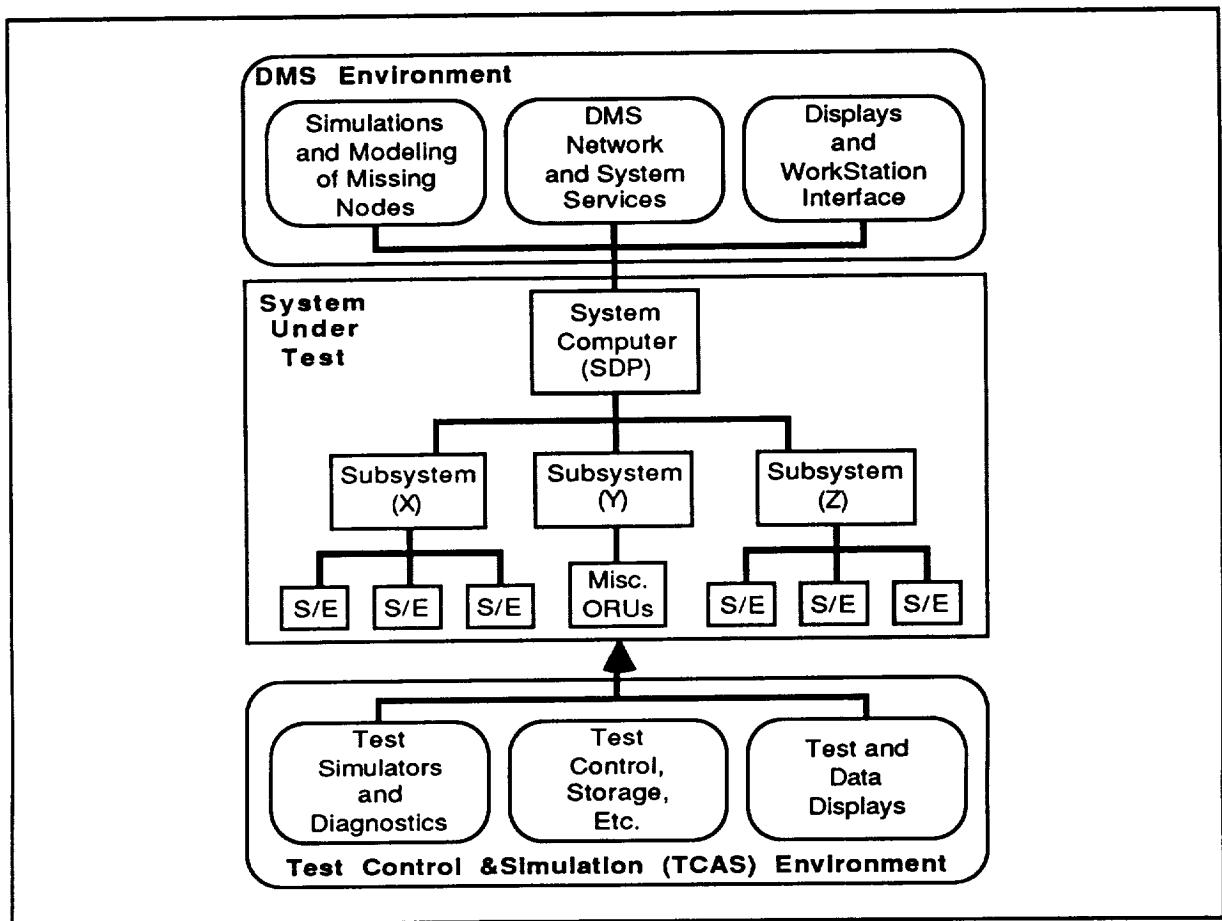


Figure 3-4. TAVERNS Concept
(Source: JSC, 1987)

Facility (SPF/SIF). The intent of DMS Kits and a common simulation environment is to provide an early ground development and test environment which duplicates (as nearly as possible) the Space Station Freedom onboard environment. The way this is accomplished in a Work Package SPF is that the SSE provides software models of systems and a Simulation Interface Buffer (SIB). The DMS Kit interfaces the system under development (e.g., a Thermal Control System) to the SPF for work package development and *intra-work package* ITV. *Inter-work package* ITV is accomplished at the MSIF.

DMS Kits will be provided in six *types*, with Type 1 consisting only of an SDP on which to run the software of the system under development/test (all DMS services are provided on the WP SPF). Starting with DMS Kit Type 3, the DMS Kit itself will provide DMS Services. To a degree (although not one-for-one), there is a DMS Kit for each of the test levels shown in Figure 3-3 (i.e., ORU, assembly, subsystem, system, multi-system integration, onboard test, etc.). Thus, DMS Kit Type 4 is the MSIF kit, and DMS Kit Type 5 is the Integration, Assembly, and Check-Out kit. Breaking this analogy, DMS Kit Type 6 is a special Payload Interface Verification and Testing kit.

3.1.3.4 Advanced Automation Capabilities

Requirements for advanced automation IT&V are preliminary at this point. LMSC (1988^b) requires that "the IT&V Element shall support IT&V of knowledge bases for artificial intelligence and expert system applications".

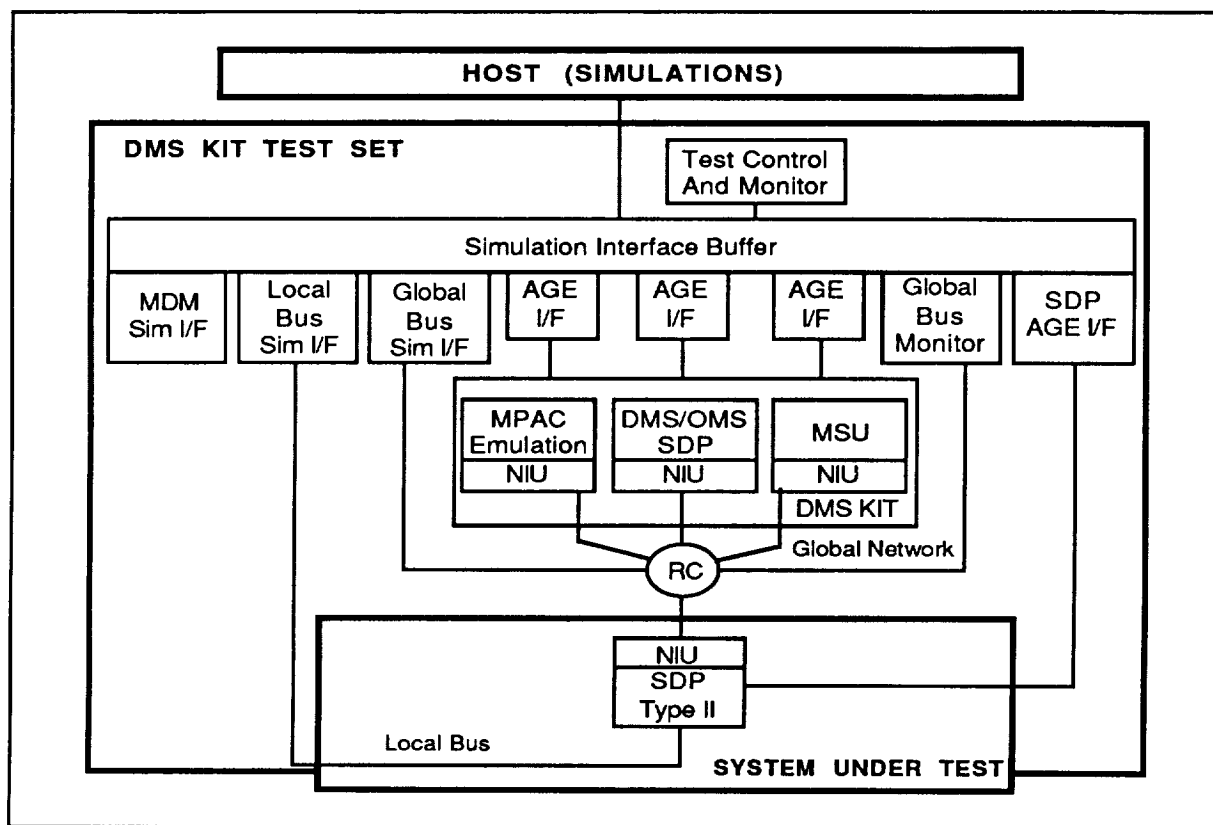


Figure 3-5. DMS Kits Block Diagram
(Source: JSC, 1987)

3.2 MULTI-SYSTEM INTEGRATION FACILITY

3.2.1 General Description

The Multi-System Integration Facility (MSIF) will be the final point for integration and test of Space Station Freedom flight software before it is certified for flight readiness. It will demonstrate that "the computational hardware/software portions of the various Space Station Freedom systems, which have been produced by different contractors at geographically distributed sites, interact correctly as defined by their designs, and meet the overall requirements of the program" (JSC, 1988^c). The MSIF will provide a high-fidelity ground-based replica of the onboard systems, using real hardware (at least up to the sensor/effector boundary) and software, or high-fidelity models, as required by the test team. The MSIF will be a physical facility, located at JSC, utilizing the rules and tools of the SSE. Its architecture is still

in the early design phases, but one can assume that it will consist of host computers and workstations like those in the SSEDf and the SPFs, along with flight-like hardware (not necessarily flight qualified) on which to host the systems under test. A special DMS Kit (Type 4) will be used to host the system under test and couple it to the DMS environment (MDC, 1988^a). The host computers will be able to run simulations both in interactive and in batch modes, in near-real time fashion. Large data bases of test cases and test results will be maintained under configuration control, with tools to manage and analyze pre-, during-, and post-test results.

The goal of the MSIF is to test and certify hardware and software systems that have *coupling* (through the DMS) with other systems. Space Station Freedom systems entering the MSIF will already have undergone thorough testing, including testing with models of other systems. In fact, the first *doorway* test of a system entering the MSIF will duplicate some of these tests. In the MSIF, the incoming system will be tested for the first time in conjunction with real interfacing systems (JSC, 1988^b). This testing will encompass both performance testing (where the new system simply presents an accurate interface load to the rest of the system) and functional testing (where the new system actually computes outputs based on its inputs).

The MSIF represents the last step in formal testing and certification of a system which is attempting to achieve flight certification. Thus the MSIF will have stringent configuration management controls, including a data base of all versions of all software (and test scripts). For systems coming to the MSIF from one of the SPFs, this strong configuration management can be viewed simply as an extension of the configuration management tools employed at the SPFs/SIFs (since the MSIF will use the SSE tools). However, for a system coming for final test and integration from an international partner (i.e., a system developed in an environment not conforming to the SSE), the MSIF may represent the first time that the system has been subject to SSE-style configuration control.

As suggested by the above paragraph, the MSIF can be viewed as adhering to the physical configuration and rules and tools of the SSE, with the addition of *real* Space Station Freedom system hardware and software (instead of, or in addition to, the baselined SSEDf system models). The MSIF is the single facility within the SSE where the integration phase of the life cycle process model must be accomplished. It should be recognized that the MSIF is not necessarily a final step in the life cycle; software that does not meet certification cycles back to SPFs and to earlier process phases of the life cycle. Such modified software will eventually need to return to the MSIF from the SPF that modified it.

3.2.2 Advanced Automation Capabilities

No special facilities or methods for integrating advanced automation software are as yet baselined. Requirements for testing advanced automation software are set forth in (LMSC, 1988^b).

SECTION 4

EVOLUTION PATHS FOR ADVANCED AUTOMATION

Sections 2 and 3 highlighted the existing state of the Space Station Freedom Program (SSFP) design, research, operational, and support capabilities as they pertain to Space Station Freedom software development, testing, and integration. Many of the necessary pieces are already in place to accommodate the goal of a flexible environment in which new development technology can be plugged-in to support the necessary software applications needed for a 30-year Space Station Freedom Program. However, a strategy for the evolution of these diverse capabilities, especially with respect to advanced automation technologies and applications, is needed. This section proposes such a strategy. The following discussions of evolution paths for advanced automation address the evolution of advanced automation technologies, the facilities which use these technologies to build software, and the applications based upon these technologies.

4.1 TECHNOLOGY EVOLUTION

This section introduces the evolution paths of advanced automation technologies within the Space Station Freedom Program, as shown in Figure 4-1. While the evolutionary end point in this diagram is the use of these technologies within onboard flight systems, the concepts can be applied to ground-based systems. It is emphasized that this section addresses *technologies*, not facilities or applications.

As can be seen in the figure, technologies evolve over time. As they move from level to level, they progressively mature from the domain of research (state-of-the-art) toward practical application (state-of-the-practice). As technologies mature, they move from technology development environments to environments where application development takes place. Products of one evolutionary level become the building blocks at the next level. For example, early technological development takes place through basic research. As research issues mature they can be used to create tools. Tools may evolve to the point where they are used to develop prototypes; prototypes to build applications; applications to build systems; systems to build Space Station Freedom.

Technologies can follow three evolutionary paths: gradual evolution, an evolutionary jump to a higher level environment, or extinction. A technology may acquire some advantage, reach a level of maturity through gradual evolution, or fill some void, allowing it to make an evolutionary jump, a leap to a higher-level environment. Of course, the technology must now compete within the new environment, with the same three paths available within this environment. A technology may become extinct, due to either some environmental change, some cataclysmic event, or a failure to compete with other members within the environment. Extinction within one environment does not mean extinction in the underlying environments.

To make a leap from one level to another, certain environmental *doorway* tests must be passed. For instance, basic research must be proven through experimentation before tool development, based upon the research, can be successful. Tools must be available and useful to prototype developers. They must perform some necessary function, either in a manner which betters the competition or fills a void. Prototypes must prove some concept, demonstrate some function, through instantiation. Applications must overcome scale-up problems usually associated with the transition from proof-of-concept prototypes to real-world applications. Performance requirements must be met; validation and verification (V&V) methodologies specific to the underlying technology must be available or created.

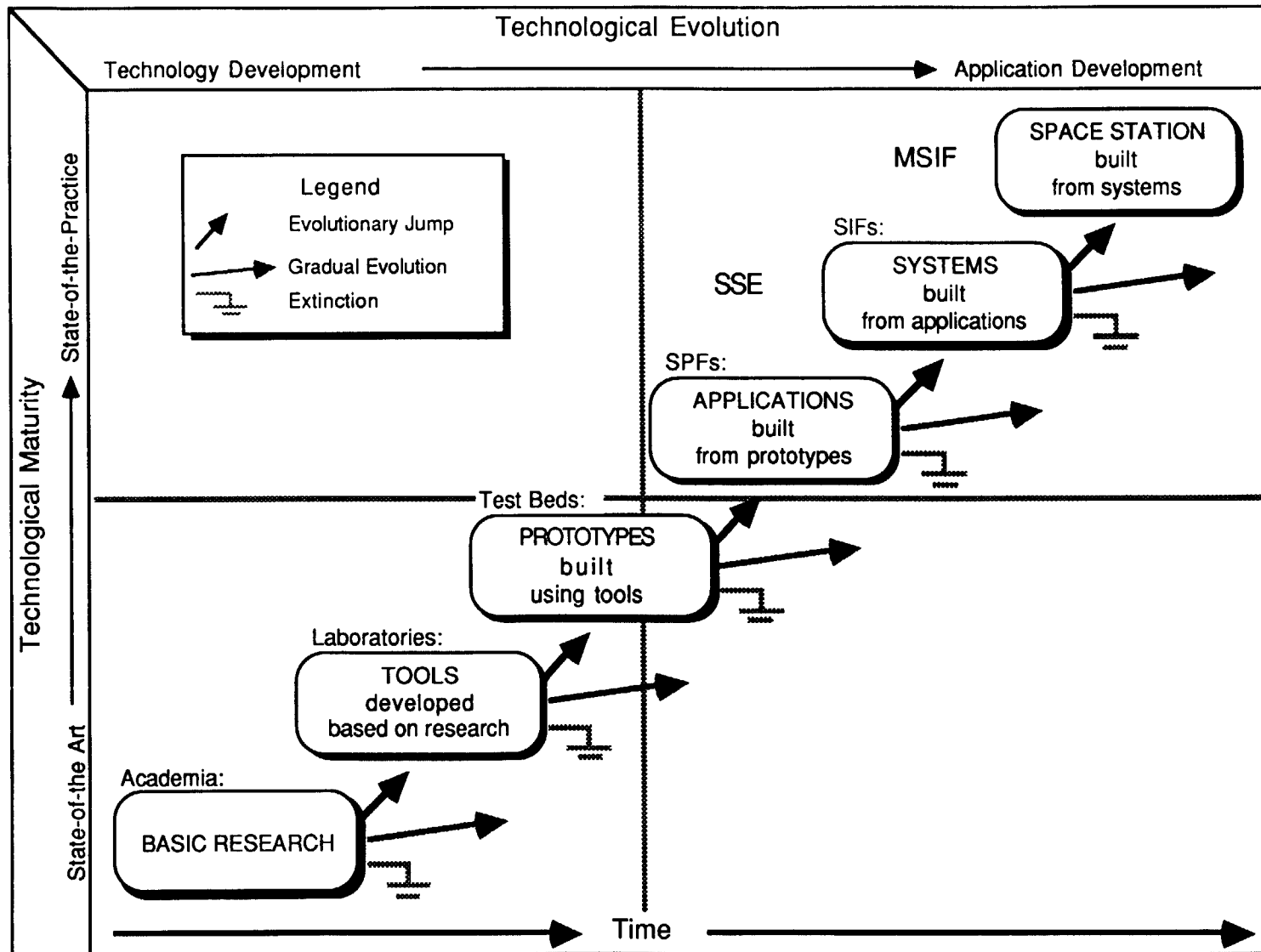


Figure 4-1. Advanced Automation Technology Evolution

Making the leap from one environment to the next does not ensure survival within the new environment. Each environmental level possesses new and different environmental factors which will act upon the technology. These new criteria will need to be satisfied if the technology is to evolve beyond a certain level. Various technologies will scale the path at different rates and at different times. Not all technologies will evolve to completely traverse the path.

Some examples might help to make these concepts clear. Neural network research flourished in the early 1960s; a large amount of government research money was awarded and it looked like a promising field. Neural network technology was mainly at the level of basic research, but tool development had begun when Marvin Minsky and Seymour Papert published their book *Perceptrons* (Minsky and Papert, 1969). In this book, they showed that neural networks had theoretical limits, at least the extant models of the time, which would restrict their use severely. Many people attribute the early demise of neural network research to the publication of this book; Minsky argues that other factors had precipitated the death of neural networks several years earlier. Nevertheless, the *classical* period of neural network research ended, and the *dark ages* began in which "research on neural networks was unloved, unwanted, and, most important, unfunded" (Anderson and Rosenfeld, 1988).

During these dark ages, neural network technology was virtually extinct in all but a small number of basic research laboratories. That is, until the *renaissance* of the 1980s when, as a result of many evolutionary jumps from new research, results to tool development to effective prototypes, real-world applications began to be developed. Neural network technologies are now available in tools from at least twenty companies; these tools are available on standard platforms, and there is some talk of developing standards. Neural network technologies are on their way to becoming *conventional*, at least at the tool level. At the same time, however, certain neural network applications will evolve slowly due to performance issues associated with serial hardware platforms. Massively parallel hardware will allow neural network technology to take a massive evolutionary leap.

Several expert system technologies have evolved to a higher level of maturity and conventionality. Some of these technologies are now widely accepted within the business and engineering worlds. These technologies are state-of-the-practice; others are still research issues. Contrast two types of expert system applications: diagnostic systems and planning systems. Edward Feigenbaum, in his Presidential Address at the 1988 American Association of Artificial Intelligence (AAAI) Conference, suggested that diagnostic applications are the most numerous expert system applications. This is because the technologies required to build diagnostic systems have evolved to the point where there are many tools available on a variety of hardware platforms which have allowed the development of useful diagnostic applications. The technologies required to develop planning systems, however, have not evolved to the same level of maturity or readiness.

Diagnostic expert systems are approaching the upper right-hand corner of figure 4-1 while planning systems still reside in the lower left on this diagram. Put another way, diagnostic systems have matured to be state-of-the-practice, while planning systems are still state-of-the-art. Diagnostic systems have made the leap from technology development facilities to application development facilities; planning systems, and other immature technologies, have not, yet.

The major low-level evolutionary drivers of expert system technologies are languages (e.g., LISP, Prolog), knowledge representations (e.g., rules and frames), and inference mechanisms (e.g., forward and backward chaining). Rule-based knowledge representation schemes and inference mechanisms are mature and useful for diagnostic systems. It is possible that planning systems will require other technologies which are still relatively immature.

Higher-level drivers of rule-based technologies, tools, are widely available from many different vendors. Users have embraced rule-based diagnostic applications and developers (i.e., the *Data*

Processing (DP) shops) are becoming comfortable with rule-based expert system development. Many successful diagnostic prototypes and applications have been developed and reported in the literature.

Specialized platforms for the development of rule-based prototypes and applications (i.e., dedicated LISP machines), at one time quite a mature technology, are rapidly being replaced by conventional hardware platforms (e.g., personal computers and minicomputers). These specialized machines are rarely seen above the prototype level on the evolutionary path. Some of the companies which created and sold these machines are extinct. The specialized machines failed to compete with the extant and emerging more conventional hardware in terms of cost, performance, and integration. It is unlikely that these machines will be used as a delivery platform for applications, especially within the SSFP. However, as Peter Friedland et al. (1988) pointed out in their report to Space Station Freedom Level I, the development environment does not necessarily have to be identical to the delivery environment. Specialized development platforms are useful for rapid prototyping and experimentation, and the job of porting applications from this environment to a more conventional delivery environment need not be cumbersome.

A major high-level evolutionary determinant is the testing required before a technology will be considered safe, reliable, and ready for use in space. Considerable work has gone into the development of V&V methods for rule-based expert systems, although a great deal of work is still needed in this area. V&V methods will be required for all emerging technologies, at least with respect to the qualities that make these technologies unique and require different forms of testing.

4.2 FACILITY EVOLUTION STRATEGIES

The preceding section discussed each level of the technology evolutionary path from research to flight systems, along with the facilities and products associated with each level. This section proposes a strategy for promoting the use of advanced automation within the SSFP.

Early thinking on the subject of the promotion of advanced automation within the SSFP suggested the need for one or more physical Advanced Automation Test Beds. These Advanced Automation Test Beds were considered necessary to fill the gap which exists between the technology developers and the application builders. The authors now reject this concept. Not only are Advanced Automation Test Beds unnecessary; they could be detrimental. Creation of one or more Advanced Automation Test Bed would hamper the evolution of the Software Support Environment (SSE) and would duplicate, or compete unnecessarily with, the existing laboratories and test beds. Advanced automation technologies will need to interface with or be integrated with conventional technologies, which will be developed in the existing facilities.

A more appropriate direction, then, would be an examination of existing facilities. These facilities, the laboratories, test beds, SSE, and Multi-System Integration Facility (MSIF), will evolve over time. They will need to evolve in parallel with the evolution of the Space Station Freedom. They must evolve to incorporate new technologies as they are born and mature. They must evolve to provide greater levels of autonomy to flight systems.

4.2.1 The Incorporation of New Technologies

The use of *new* technologies is usually resisted. Friedland et al. (1988) report that the only real resistance they observed to the immediate use of Knowledge-Based Systems (KBSs) on Space Station Freedom came from the "institutional" Management Information System (MIS) community at NASA. This was not a surprising find, either to those authors or to us. Resistance from the DP shops is

sometimes a stumbling block to the use of these new tools; at other times this resistance buffers users from untested, immature technologies and unsafe operations.

Researchers are always hanging over the cutting edge of technology, entrenched in the state-of-the-art. The engineering world is based upon proven technology, but at the same time is always investigating new technologies that survive the research laboratories. Users and operations groups want state-of-the-art. These are the major drivers pushing new technologies toward application development, sometimes too early.

Developers (i.e., the DP shops), on the other hand, are entrenched in the state-of-the-practice, especially in a contractor-oriented software development effort where budgets may be constrained. If something can be done the old way, why change? If money can be saved building the Space Station Freedom without advanced automation, why include expert systems? Tunnel vision often restricts progress too much, but it also occasionally averts danger.

The first use of new tools yields fragile systems; they do not perform well, they seem hard to use, they require retraining, etc. However, as the tools are refined and as system developers become more proficient in their use, the systems become increasingly more robust. Some tools never produce robust systems, and they eventually die along with their aberrant systems. The authors use the word *aberrant* deliberately; to be aberrant means to depart substantially from the standard. As a building block matures, a true test of its ability to survive is the eventual formation of a standard for the underlying technologies and tools (perhaps a new standard, which legitimizes the tool and its resulting systems). Out of these standard tools, reliable and cost-effective systems can be built.

4.2.2 The Need for Two Development Environments

The technology evolution path is divided into two distinct segments: state-of-the-art technology development and state-of-the-practice application development, as shown in Figure 4-1. Therefore, two separate advanced automation development environments must exist within the SSFP, but they must fit within the established framework.

4.2.2.1 Laboratories and Test Beds: State-of-the-Art Technology Development

Evolution of Space Station Freedom is critically linked to the laboratories and test beds. New technologies will be nurtured within these facilities, and new tools will be developed which will allow these technologies to be used in flight systems. However, new technologies should not be used until they achieve a certain level of maturity.

The technology levels and associated facilities which are concerned with state-of-the-art technology development are shown in the lower left-hand portion of Figure 4-1. Many of these facilities exist today, preceding the SSE by several years, and are actively testing critical design, operations, and integration concepts. Some of these facilities will gradually disappear; many others will continue to research and test new concepts and designs throughout the life of the SSFP. In some cases, these facilities will produce useful applications based on technologies which are not yet considered fully mature or conventional. This does not imply that these applications should not be used. If they solve a problem which cannot be solved using more mature technologies, these applications should be utilized; however, they will probably require a greater level of scrutiny.

4.2.2.2 SSE: State-of-the-Practice Application Development

The long-term success of the SSFP is critically linked to the SSE *factory* that will supply Space Station Freedom software and to the infusion of new technology. If new technology does not continually drive this factory, it will become obsolete and cost-ineffective.

The SSE facilities, shown in the upper right-hand portion of Figure 4-1, are responsible for application and systems development. The SSE should be restricted to the use of state-of-the-practice, conventional technologies. The SSE, regardless the level of its flexibility, is not a satisfactory environment for the development and test of state-of-the-art advanced automation applications. State-of-the-art implies technological immaturity, limited developmental and pragmatic experience base, and the lack of standards. These technologies should be left to the research laboratories and the test beds. With suitable flexibility, the SSE can accommodate state-of-the-practice advanced automation applications. Of course, in an evolving world, today's state-of-the-art is tomorrow's state-of-the-practice.

What is needed is a set of criteria and procedures to promote and control the evolution of the SSE, to restrict the use of immature technologies within the SSE, and to allow the SSE to include new technologies as they mature.

4.2.3 Implications for the Laboratories and Test Beds

As proposed above, only well-established tools can be placed in the SSE for use in developing safe and reliable advanced automation applications. The places for emerging advanced automation tools and applications are the SSFP design test beds and laboratory facilities since new advanced automation technologies, tools, prototypes, and applications will emerge from these facilities.

4.2.3.1 Research Laboratories

NASA efforts such as the Office of Aeronautics and Space Administration (OAST) System Autonomy Technology Program (SADP) suggest the value of laboratories in bringing advanced automation to market. Such programs (and others within the NASA center and Work Package structure) are the Research & Development component of the SSFP organization. The SATP Plan (ARC, 1987^a) sets forth five technology areas (task planning and reasoning, control execution, operator interface, sensing and perception, and system architecture and integration) where NASA feels that R&D is required to bring advanced automation and autonomous systems to fruition.

One of the goals of the laboratories is tool development. Investigations into new technologies will produce useful tools as these technologies mature. Migration of these tools from the laboratories to the test beds and, subsequently, to the SSE will allow the development of prototypes and applications based on mature technologies. Therefore, criteria and procedures must be defined for determining the technology readiness of the various tools developed before they can emerge from the laboratories and be used within the test beds and the SSE.

4.2.3.2 Design Test Beds

Laboratories can be ad hoc, but test beds need more structure and organization. This is because test beds are dealing with engineering and operations personnel, and because they are the step just before application migration to the SSE. Test beds are essential to the engineering component of the SSFP organization. Prototyping activities are important for the testing of critical designs and operations concepts. These facilities will provide essential ground capabilities as Space Station Freedom evolves.

Test beds are also essential to the operations organization. Training of and experimentation by operations personnel may take place within system test beds. In other words, flight controllers would not know what to do with an Operations Management Ground Application (OMGA) (or would be afraid to use it) if they had not already used an analogous capability on an engineering test bed.

One of the goals of laboratories is *tool* development and migration of these tools to the test beds and the SSE. An analogous goal for test beds is *prototype*, and in some cases, *application* development, with subsequent migration to the SSE for reimplementation or directly to the MSIF for integration (assuming the application has been designed and developed for testability and maintainability). Just as there is a hierarchy of tool readiness within the laboratories, there is a hierarchy of application readiness within the test beds. Thus, it could be argued that diagnostic systems are ready to migrate out of test beds to the SSE, but that scheduling and planning systems are not (tools for building them are still evolving in the laboratories). A mechanism is required to allow prototypes and applications to be evaluated for technology readiness; a doorway test to gain entrance either to the SSE as a viable, mature technology to be supported by the SSE; or to the MSIF as an important application for the Space Station Freedom, using a new, less mature technology.

The OAST has embarked upon several technology demonstrations to show the state of readiness of some of the underlying core technologies as well as the interaction of advanced automation applications on distributed systems. These demonstration and integration activities involve the integration and coordination of several advanced automation laboratories and design test beds.

In a like manner, the End-to-End Test Capability (ETC) integration activities require the coordinated efforts and integration of many test beds to test operations concepts and critical designs. These activities represent an early level of demonstration, test, and validation that will not be available anytime soon in the SSE world. The ETC can play an essential role in the current and ongoing station design and development efforts. A similar Strategic Defense Initiative (SDI) activity exists within the National Test Bed. The Defense Department's Strategic Defense Initiative and Air Force Electronics Systems Division are developing this test bed which will be a high-fidelity, distributed simulation capability for testing and developing software and systems architectures for SDI. They have created a 12-member advisory group of scientists called the Simulation Engineering Panel. Dr. James Brown, professor of computer science at the University of Texas at Austin, is the panel's chairman. He has an extensive background in operating systems, performance evaluation, parallel processing, and supercomputing. Dr. Brown believes that the technology underlying the SDI test bed could be a valuable tool to NASA (Gruman, 1988).

The interconnection, integration, and coordination of existing and planned SSFP test beds is critical to the development of a safe, reliable, and effective Space Station Freedom Program. A mechanism must exist, however, that facilitates two way communication between these facilities to allow cooperative problem solving.

4.2.4 Implications for the SSE

Core technologies, tools, and applications can be grouped according to technology readiness and evaluated for support by the SSE. The SSE should use tools and develop applications only if the underlying technologies are mature and conventional. Certainly rule-based knowledge acquisition and representation, window-based human-computer dialog, and certain knowledge-based management architectures have entered the mainstream of the computer industry. The underlying tools of these and other core technologies should be included within the SSE. Tools for less mature technologies should remain in the laboratories and test beds. Similarly, only those applications which use mature technologies should be considered for the Initial Operations Capability (IOC) of Space Station Freedom.

A goal of the SSFP, then, should be the identification of these mature, enabling technologies and applications; and a plan should be developed for selecting, developing, and migrating appropriate tools and techniques out of the laboratories to the SSE. This migration will not be painless; it will take education, training, and persistence in the face of resistance. But, it is necessary, because the SSFP will require these applications to meet its system autonomy objectives. The SSE will be responsible for moving mature advanced automation applications (prototypes) from a demonstration environment to an operational environment. These applications must be re-developed within the SSE, and this requires tools and training in the development environment.

The Software Support Environment Development Facility (SSEDF) is the element of the SSE that provides the tools that the SPFs will use for software production. The SSEDF can be useful in providing tools to bring advanced automation applications to flight readiness. A list of the suitable advanced automation applications which it can support must be developed. Such applications, of course, represent a moving target. Friedland et al. (1988) suggest a group of candidate applications for Space Station Freedom IOC, and another group for beyond IOC. In the same paper, the authors suggest SSEDF tools to bring the IOC applications to market. The suggestions include the development of an *official* Ada expert system shell within the SSE, the acceptance and implementation of a life cycle model (e.g., the Boehm spiral model) that is appropriate for advanced automation development, and the adoption of methods for KBS V&V. In the following sections we will consider these recommendations in terms of their impact on the already baselined SSEDF.

4.2.4.1 An SSEDF Expert System Shell

Several commercially available tools, expert system shells, should be considered as constituents of the SSEDF toolset, as opposed to the creation of an SSEDF standard Ada shell. Of course, the characteristics which would allow this Commercial off-the-shelf (COTS) shell to be included in the SSE, the *doorway test*, would need to be identified before this or other similar selections could take place. What are some of these criteria that would define the product as being *conventional*?

First, a COTS product must be well established and accepted within the marketplace. It must have: a suitably long history of use; a large base of applications developed with the product; a large community of developers who have used it (and been trained in it); and a large community of users who have successfully used applications produced by the tool. Secondly, the product must run on a variety of conventional platforms under conventional operating systems. Of course, these platforms must be the same as the ones chosen for use within the SSE, not necessarily the same ones certified for flight systems. The product must be integrated with mainline application products, ideally using standard interfaces between or among these products. It must be backed by a strong company, either with strong internal strengths, or with cooperative marketing arrangements with mainline vendors.

An important milestone for any technology is the development of a standard related to that technology. To date, there are few, if any, advanced automation standards. It is anticipated that standards for certain advanced automation technologies (e.g., rule-based knowledge representation and inference) will be developed in the near future.

Some advanced automation technologies may have reached a level of maturity compatible with their incorporation into the SSE. Certain products which use these technologies may meet the criteria of the SSE doorway test. Less mature technologies may also be included as functions of these products, but this may not cause any problem if the use of these functions is restricted.

Is an Ada-based expert system shell a necessary development tool within the SSE? Probably not, especially if a commercial tool meets the SSE expert system tool requirements. The delivery of applications

to flight systems may require a special expert system shell which will reside on flight hardware and meet performance requirements.

4.2.4.2 Life Cycle Models

There is no single life cycle model that is suitable for all classes of users and all application domains; each has its strengths and its weaknesses (see Appendix B). Because the traditional waterfall life cycle model is the basis for current software acquisition standards, it might seem to be the appropriate choice for the management of advanced automation software development for the sake of commonality. However, it is generally agreed that the waterfall model is applicable only to the category of advanced automation software for which most of the knowledge is acquired from documented sources so that complete specifications for a system can be derived. Only a small percentage of KBSs fall into this category. The transform life cycle model offers the advantages of automatic code generation, but it is difficult to apply in this domain because, like the waterfall model, it is specification-based. Furthermore, the application of the transform model to the development of advanced automated software is still in the research stage.

Both the iterative model and the spiral model are suitable to manage the development of software with the special characteristics of KBSs. The iterative model is the most frequently applied life cycle model because it supports applications with initially ill-defined interfaces and functions. This model easily provides for the incorporation of expert knowledge and user experience into an iteratively refined product. The spiral model is an appropriate model for KBSs because software developed using new and evolving technologies is best managed and controlled by a life cycle model that directly addresses risk analysis at repeated intervals throughout the development cycles.

It is recommended that the iterative development model augmented by panel reviews and by further extensions from Barry Boehm's risk-driven spiral approach be adopted for the management of the development of KBSs. Each of the phases identified in the iterative life cycle model can be considered as a cycle of a spiral, each cycle having essentially the same phases, but with increasing levels of elaboration. This proposed spiral/iterative development life cycle model would apply a management structure to the development of advanced automation systems that is flexible enough to support the changes that will occur with new technologies as well as the changes that are an inevitable part of the refinement and evolution of a KBS. These life cycle recommendations are not in conflict with the recommendations of Friedland et al. (1988).

Panel reviews would provide continuing requirements documentation, user involvement at all stages of development, and software verification at the end of each phase. The functions of the panel should be expanded to include the risk analyses described by Barry Boehm (1988) in his recent article in *IEEE Computer* in addition to the functions described by Chris Culbert et al. (1988), as presented more fully in Appendix B.

Many of the milestones, as well as the critical requirements and test plan documentation, of the traditional waterfall life cycle model are provided for with this recommended management approach. The initial requirement review (IRR), preliminary design review (PDR), and system design review (SDR) fit within this management structure. The panel consolidates requirements into a formal System Requirements Document (SRD) that evolves as the KBS is developed iteratively. A test plan and a library of test cases evolve during system development. After the KBS passes testing, it can be put under standard configuration control. Maintenance can be handled in a standard manner, but extensive changes may require a return to the beginning of the development life cycle.

4.2.4.3 KBS Verification and Validation

V&V are important activities during all stages of software development, therefore, appropriate V&V methods will need to be developed for each life cycle phase. The development of V&V methods for new technologies will be required as these technologies mature and move into the SSE. This does not imply that existing testing methodologies will not be applicable. It will be the characteristics of these new technologies which are somehow different from other technologies that may require new V&V methods.

Since KBSs, especially rule-based systems, are the most mature advanced automation technology, have the largest user and application base, and will probably be the first of these technologies to be included the SSEDf toolset, specific attention to V&V methods for these systems should be developed first. The KBS V&V literature is surprisingly large. There are many theoretical papers and probably an equal number of pragmatic papers, but the field is a long way from having proven methodologies.

The authors agree with the view of Friedland et al. (1988), and several other authors, that it is a myth that V&V of KBSs should be extremely difficult or even impossible. The challenging aspects of developing V&V methodologies specific to KBSs, and other advanced automation systems, will probably relate to the behavior of the system. These systems attempt to emulate human decision making, and human inferencing in general, usually in the operation of complex processes. These systems often deal with problems where more than one correct answer can be generated. Often, human experts will disagree about which answer is the right answer. V&V methodologies for these systems will need to determine not only if the answers generated are correct, but also whether or not they were determined in the proper manner. (It is possible that the correct answer could be generated for the wrong reasons.)

These characteristics will present validation challenges and have little to do with implementation issues. Testing these behavioral characteristics will probably be very similar to testing humans. Experts will be closely associated with the examination of these behaviors. The problem of validating the expert, proving the worth of the expert, will need to be dealt with. Also, it will be necessary to prove that the transformation from expert knowledge to code (i.e., rules and facts) was performed correctly.

On the other hand, many aspects of these systems will present fairly common verification problems. For instance, the process of verifying that an inference engine was built in a proper manner and performs correctly can utilize conventional testing methods. Static code checkers (in this case rule checkers) have been developed which check for completeness and consistency, as well as logic errors, within knowledge bases.

4.2.5 An Organizational Model for Technology Development Evolution

To accomplish the goals of the Office of Space Station- (OSS) and OAST-supported technology development and test bed integration activities, open communication, technology transfer, and coordination between these facilities is critical. Most of the laboratory and test bed personnel who were interviewed requested information from us concerning the activities at other centers, test beds, and laboratories. Many of these people could not identify an existing communication channel for obtaining this kind of information. The technical success of these activities depends upon a management approach to solving these communication and coordination issues. The authors propose a strawman management process that will facilitate the dissemination of critical information.

Within this process, appropriate technology areas would be identified and key personnel would be selected to participate. An integration team from each technology area would be established, perhaps three individuals, with one serving as the lead person within each technology area. The responsibilities of each integration team might include activities similar to the following:

- Setting up methods for disseminating the technology knowledge base to all members of the technology group (e.g., technical exchange meetings, periodic reports)
- Setting up a peer review process for projects involving the technology, related to the appropriate application of the technology
- Providing a management individual with immediate feedback on projects which involve either poor use of the technology or resources
- Reviewing and evaluating all proposals within the technology area as a part of the normal process of selecting projects
- Establishing hardware and software standards to maximize interoperability and resource sharing among laboratories and test beds
- Assuring that proposals to common customers (e.g., facilities, systems, elements) are coordinated
- On a six month basis, setting up with appropriate management a two hour review to discuss
 - Status of projects or capabilities
 - Shortfalls in capabilities
 - Recommendations

4.2.6 An Organizational Model for Application Development Evolution

How can advanced automation tools be brought into the SSE as they mature and at the same time have the use of immature technologies be restricted? The authors suggest an organizational model that has served the development communities in several progressive organizations (both commercial and government) very well. With some variations from the ideas proposed here, this model may serve the SSE equally well. This model is sensitive to the following attributes that impact the acquisition and use of new technology:

- Development community (i.e., SSE) readiness or awareness
- Advanced automation team approval
- Dollar signature limits
- Procurement cycles

4.2.6.1 Development Community Readiness or Awareness

An important precursor to the development of any system for regulating the use of advanced automation technologies within the SSE is a review of the current SSE requirements and the augmentation of these requirements to include necessary capabilities and evolutionary hooks and scars.

Several years ago, at the time that the requirements and the Request for Proposals for the SSE were being prepared, KBSs were lumped with other Artificial Intelligence (AI) technologies. They were considered subjects for research laboratories with development and deployment on stand-alone, single-user, special purpose platforms. Since the development of early KBSs generally was viewed as ad hoc, one of a kind, and difficult, the policies, procedures, and tools to support KBSs are not included in the interim environment or the initial environment of the SSE.

KBSs have moved from special purpose hardware to conventional platforms such as will be part of the SSE and flight systems. Many commercial off-the-shelf (COTS) products are marketed to support the

development and deployment of these KBSs. As KBSs have moved out of the research world, there has been growing recognition of the importance of integration, portability, fielding, testability, maintainability, and documentation in the design of KBSs. Software engineering is now being accepted as applicable to the development of a KBS that is carried through to delivery and maintenance (see Appendix B). When software engineering is applied to KBSs, these systems can be judged, like conventional software, by their long-term reliability and performance in end-user environments. Tools and methodologies for V&V of KBSs are being developed.

The MITRE Corporation recently prepared a proposal, through the Engineering Directorate at JSC, to support the Artificial Intelligence, Expert Systems, and Technology Working Group (AIESTWG) on an SSE-related task. The objectives of this task are the following:

- Define the extensions to the SSE which will support the life cycle of KBSs
- Define the set of policies, procedures, and tools to be supported by the SSE to aid the development and maintenance of KBS applications
- Define program-wide standards related to the software engineering and management of KBSs, in order to provide support of inter-operability between Space Station Freedom participants, commonality, V&V, and to reduce life cycle costs

The proposal included examples of tasks which might be done to accomplish these objectives including the following:

- Definition of additional SSE standards and tools for the support of KBS throughout the software life cycle
- Evaluation of the impact of adding the proposed standards and tools to the SSE
- Evaluation of the impact of the proposed standards and tools on the users of the SSE

A similar study is required that would establish the technology readiness of advanced automation technologies and recommend a list of technologies to be included in the SSEDf toolset.

4.2.6.2 Advanced Automation Technical Review Team

A team made up of technically strong members would provide a first-level review of a request for an advanced automation tool. This team associates the request with one (or more) of a set of categories (these categories need to be defined) that apply to advanced automation tools and determines the parameters associated with the categories. For example, a request for an expert system shell would be compared against the set of *legal* expert system shells (i.e., those that have a stable organization behind them, those that have been used before in the organization, those that are compatible with the hardware and operating system software within the organization, etc.). The underlying technologies used within the shell would be compared with a list of the *legal* technologies. Approval for the acquisition and use of the new shell would be determined based on these lists of legal entities, or a review of any new technologies requested, and a judgement concerning their maturity.

But this team would do more than approve advanced automation procurement requests. It would also perform the following activities:

- Set up methods for determining which advanced automation technologies and tools are *ready for use* within the development community
- Review technologies as they emerge from the SATP and SSFP test beds, as well as other programs and facilities, for *readiness*

- Periodically brief senior managers on the status of advanced automation tools, and recommend ways to develop tool awareness in the SSE community

4.2.6.3 Dollar Signature Limits

This attribute of the review team's responsibilities recognizes that the acquisition of tools (hardware or software) below a certain dollar level are more-or-less exempt from the acquisition process. The limit varies with organization, usually in the \$1K to \$10K region. It has allowed organizations to purchase a wide variety of PC-based AI hardware and software over the past several years, simply on the signature of one responsible individual. Such tools have been used for experimentation and prototyping, or in some cases (like the PC revolution) have provided the means of bypassing mainline ways of doing business. The latitude offered by applying this rule is a necessary and desirable part of the SSFP development process. Of course, conflicts with the list of *legal* SSEDF tools and technologies would need to be resolved.

4.2.6.4 Procurement Cycles

When the dollar limit imposed above is exceeded, organizational personnel usually enter a formal procurement process, normally involving more signatures and the awareness or scrutiny of the procurement organization (e.g., a purchasing department). Different paths through the procurement process are usually available; the SSE requires a special process for the acquisition of advanced automation tools. This special process requires the approval of the review team.

4.3 APPLICATION EVOLUTION

This section summarizes the preceding discussions as a scenario for the evolution, or movement, of an advanced automation application through the various SSFP facilities on its way toward use in flight. Many paths are possible, as shown in Figure 4-2. Which of these paths will be considered *legal* is uncertain at this point.

4.3.1 Application Development

4.3.1.1 Pre-SSE Application Development

Prototypes are currently being developed within the various OSS and OAST supported laboratories and test beds. It is possible that many of these application prototypes will mature to the point where they can provide important functions to the SSFP before the SSE possesses the capabilities necessary for the development of applications using the underlying technologies. If this is true, then it is equally possible that some of these prototypes will become utilized as applications without passing through the structured world of the SSE. However, the same attention to configuration control and design for testability and maintainability, which will be fostered by the SSE, must be applied to any deliverable products developed outside of the SSE.

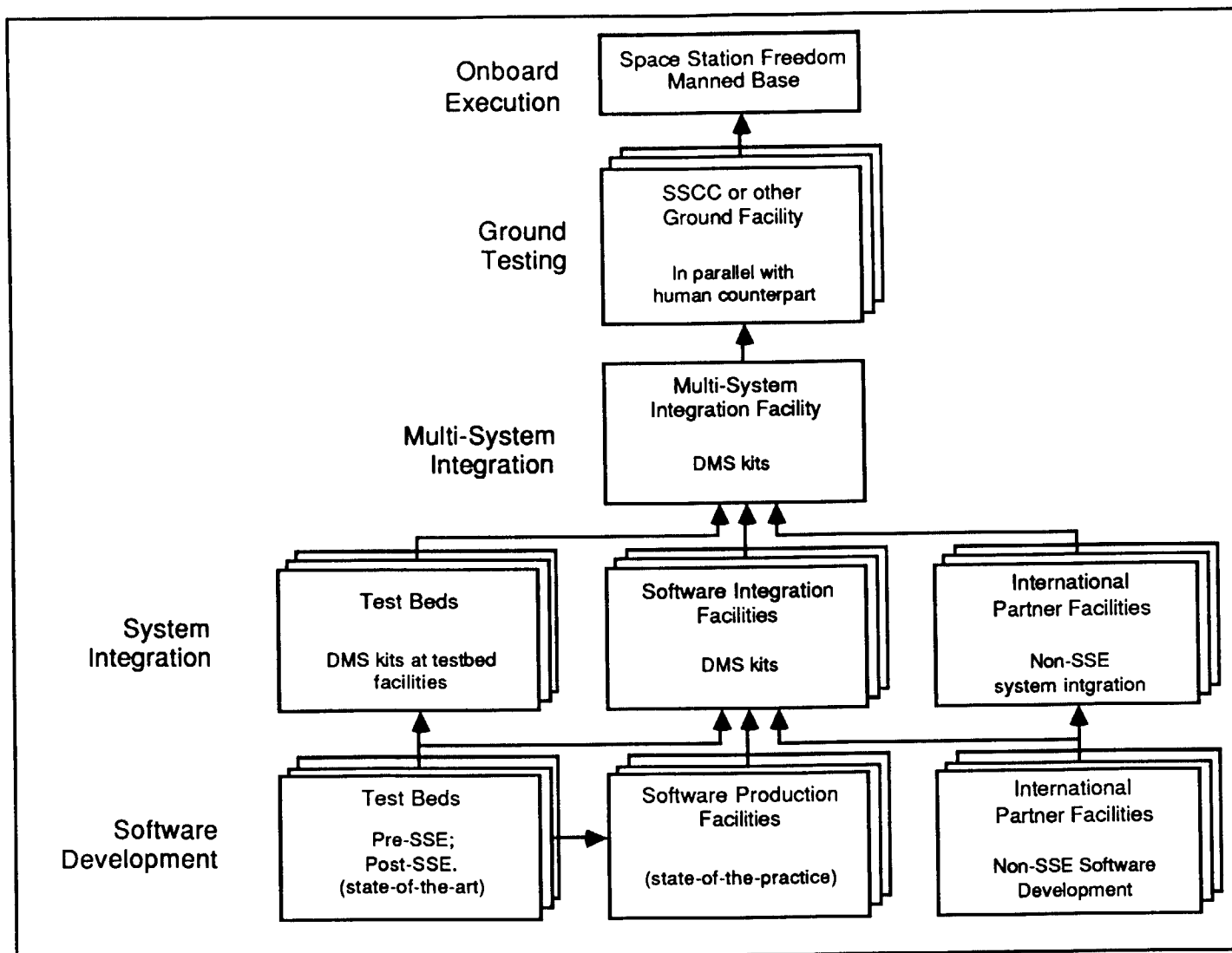


Figure 4-2. Possible Application Evolution Paths

4.3.1.2 Post-SSE Application Development

Once advanced automation tools and services are available within the SSE, there are three possible paths for advanced automation software development. Applications with well-defined requirements that depend upon mature, conventional technologies supported by the SSE should be developed within the structured and constrained environment of the SSE (i.e., within a SPF). Other, more state-of-the-art, applications may be developed within the test beds. A third path involves applications developed by international partners outside of an SSE-supported environment.

4.3.2 Intra-System Integration

The integration of applications within a particular system could take place within the SSE (i.e., on DMS-kits within a SIF), within test beds (probably using DMS-kits), or at international partner facilities which do not include SSE-supported hardware or software.

4.3.3 Multi-System Integration

Multi-system integration, at least with respect to NASA systems, will take place entirely within the MSIF. The integration of systems within international partner modules is an open issue.

4.3.4 Ground Testing

Appropriate testing on the ground is an important precursor to moving an application from the ground to residence and execution onboard Space Station Freedom. A major part of this testing, subsequent to normal V&V activities, should be parallel operations within the Space Station Control Center (SSCC) or another ground control or support center, such as a Transition Flight Control Room- (TFCR) like facility. This type of testing was used as a part of the INCO Expert System Project (IESP) within the Mission Control Center (MCC) during the flight of STS-26. The IESP hardware and software execute beside the Integrated Communications Officers' regular console so that its response to nominal and contingency operations can be compared with other conventional methods already in use in the MCC.

This type of testing allows the system to be tested in an environment with fewer constraints than would be imposed by onboard testing. These constraints include safety, performance, size, crew availability, power, and probably many others. Another benefit of this type of testing is the capability to test small pieces of specific systems as they are developed rather than testing the entire system in space. Engineers can test design issues, ground controllers and astronauts can evaluate operations concepts, before the system is used in space.

The appropriate amount of time required will vary with each system. It is possible, and advisable, to use many applications on the ground for long periods of time. Many applications could reside permanently on the ground and never have to be limited by the constraints of onboard use. One goal of automating flight systems is to reduce the number of ground personnel required to support flight operations. In many cases, this goal can be met with either ground or flight applications.

4.3.5 Onboard Operations

Of course, there are constraints that will require certain applications to reside onboard the Space Station Freedom. These constraints include safety issues, truly autonomous systems operation (in the sense of being independent of ground operations), timing requirements, and limited communications bandwidth.

4.4 SUMMARY OF EVOLUTION ISSUES AND STRATEGIES

As advanced automation technologies mature and standards are defined, reliable and cost-effective systems can be built using these new technologies if the laboratories, test beds, SSE, and MSIF are prepared to support their application. Currently, the most mature of the new technologies is KBSs. No longer are KBSs considered magic; they are just a new kind of software. The development activities for KBSs correspond in an approximate way to those performed in a conventional software development project, although they are performed in a somewhat different fashion. To support these differences, existing facilities need not be replaced; they need merely to evolve to support these new technologies.

Two development environments must exist within the SSFP to promote advanced automation technologies to the required level of technology readiness: environments for state-of-the-art technology development and state-of-the-practice application development. For the SSFP, technology development is nurtured in laboratories and test beds; application development is supported in the SSE. Evolution of Space Station Freedom is linked both to the laboratories and test beds and to the SSE software factory. These existing facilities should evolve technologically in parallel with the evolution of the Space Station Freedom.

The following evolution issues relating to the existing facilities have been identified and discussed in this review of SSFP capabilities for the promotion of advanced automation:

- A mechanism is needed that facilitates two way communication between facilities to allow cooperative problem solving, interoperability, integration, and coordination of existing and planned SSFP laboratories and test beds.
- Criteria and procedures must be developed for determining the technology readiness of tools before they can emerge from the laboratories to the test beds and subsequently to the SSE.
- A mechanism is required to evaluate commercially available tools and test bed developed prototypes for technology readiness; doorway tests to gain entrance either to the SSE or the MSIF must be defined.
- SSE support for a spiral/iterative development life cycle model is required for KBS applications, where sufficient knowledge is not available for the complete specification of requirements. Specifications are required early in the life cycle of the traditional waterfall life cycle model.
- Development of V&V methods are required for the characteristics of KBSs that make these systems different from conventional technologies. The problems of validating the expert and the proving that the transformation of expert knowledge to code (i.e., rules and facts) must be addressed.

A management approach is required to respond to many of these evolution issues and to manage facility evolution to accommodate emerging technologies. An integration team with a small number of members from the new technology area of KBSs is the proposed management model. As new advanced automation technologies are identified as candidates for the SSFP, additional integration teams would be established. An integration team would have the responsibility to establish methods for disseminating the technology knowledge base of that technology; review and evaluate proposals within the technology area; establish hardware and software standards to maximize interoperability and resource sharing among laboratories and test beds; coordinate proposals to facilities, systems and elements; perform peer review of projects involving the technology; and provide feedback to

4.5 FUTURE EVOLUTION PATH ACTIVITIES

The information, related analyses, and contacts established in the preparation of this report on SSFP capabilities for the promotion of advanced automation provide a sound foundation for further efforts in defining a comprehensive SSFP evolution plan. The objectives of the next phase of this task are viewed to be the definition of evolutionary goals for SSFP test beds and facilities, and the definition of evolutionary paths and plans for reaching these goals. In addition, collective evolutionary goals will be defined for these facilities as a set.

Before the definition of evolution plans, the endpoints (i.e., goals) must be clearly defined. Among the issues to be addressed in determining these goals are distinguishing criteria, roles and functions; interoperability, commonality, and integration; and the development, demonstration, and delivery environments. Once the destination has been well defined, the fundamental tools needed to get there must be identified. A general road map for SSFP decision making is needed in addition to test bed-specific evolution plans. Information from the Transition Program evolution studies as well as system, test bed, and contractor management will be used to tie these facility evolution plans closely with plans for the evolution of Space Station Freedom's systems. Another necessary integrating activity is involvement in the SSE and MSIF requirements definition process and the development of evolutionary plans for the SSE and MSIF relative to advanced automation.

APPENDIX A

ADVANCED AUTOMATION

Sections A.1 and A.2 briefly introduce advanced automation applications and technologies. Readers familiar with this material might want to skip these sections and continue reading with Section A.3, Candidate Applications for Space Station Freedom; and Section A.4, Advanced Automation Technology Readiness.

A.1 APPLICATIONS OF ADVANCED AUTOMATION

Hayes-Roth et al. (1983) present a list of the types of applications for which advanced automation is applicable, as shown in Table A-1. Advanced automation has been applied successfully to many problem domains including medicine, geology, aeronautics, and manufacturing, to name just a few.

Table A-1. Categories of Advanced Automation Applications

Category	Problem Addressed
Interpretation	Inferring situation descriptions from sensor data
Prediction	Inferring likely consequences of given situations
Diagnosis	Inferring system malfunctions from observables
Design	Configuring objects under constraints
Planning	Designing actions
Monitoring	Comparing observations to planned vulnerabilities
Debugging	Prescribing remedies for malfunctions
Repair	Executing a plan to administer a prescribed remedy
Instruction	Diagnosing, debugging, and repairing student behavior
Control	Interpreting, predicting, repairing, and monitoring system behaviors

A.2 ADVANCED AUTOMATION TECHNOLOGIES

Over the past two decades, many advanced automation technologies have been born and several have matured to a degree that greatly enhances the productivity of humans. These technologies can be classified as providing two primary functions: 1) simulation of human cognitive processes and 2) human-like interfaces to computer applications. The following sections introduce advanced automation techniques and interfaces.

A.2.1 Expert Systems and Knowledge-Based Systems

Expert systems are one of the first commercially viable technologies to emerge from the artificial intelligence (AI) labs of the nation's universities and research centers. According to the Advanced Technology Advisory Committee (ATAC-, March, 1985):

Expert systems is a subfield of artificial intelligence concerned with developing computer programs that use knowledge and reasoning techniques in specific domains to emulate the decision processes of human experts.

Expert systems attempt to emulate human expertise. Knowledge-based systems (KBSs) are similar to expert systems except they do not specifically model human expertise. Nevertheless, the two names are used synonymously in this report. KBSs are the most mature AI technology. They are heavily used within industry and the government. In addition, KBSs, especially rule-based systems, are the most likely candidates for early use of advanced automation applications onboard Space Station Freedom. For these reasons, this report concentrates mainly on KBS technologies. It introduces and discusses other advanced automation technologies: artificial neural systems, fuzzy logic-based systems, natural language understanding, continuous speech recognition, speech synthesis, and vision and image processing.

KBSs are highly suited for dealing with ad hoc or inexact information. They define knowledge about a particular problem domain using heuristics as opposed to specific algorithms. Heuristics are usually thought of as *rules of thumb*. Heuristics guide the program to a reasonable solution using qualitative classifications to form intermediate/hypothetical conclusions as opposed to calculating the exact solutions with some rigorous mathematical formula. This is akin to the way human experts solve problems. Humans have little capacity to perform complex mathematical calculations, but are very good at problem abstraction and pattern recognition. However, KBSs do not preclude the use of algorithms; in fact, an algorithm may be employed at any time during execution to enhance the problem solving task. Thus, KBSs are capable of utilizing all the power typically associated with computers and conventional algorithmic software in addition to emulating the way humans solve problems.

KBSs have typically been considered decision support tools not intended to replace humans but simply to augment and enhance the decision making process (i.e., a consultant). A KBS typically provides advice that may be accepted at the discretion of the user. A key feature of most KBSs that differentiates them from conventional programs is their ability to explain the path of reasoning used in coming to a conclusion. This self-knowledge is very important since the operator of most decision support systems likes to know why a response was given rather than just accept it as the absolute conclusion.

KBSs can overcome some problems that human experts are prone to encounter. The exhaustive nature of problem solving in expert systems ensures that remote or obscure possibilities are not overlooked. It is very difficult for a human to consider all possibilities; in fact, humans have a tendency to filter out information that may not appear to be relevant. KBSs eliminate possibilities systematically only when they are considered absolutely irrelevant and retain all other possibilities until they have all been investigated. In many cases, expert systems have appeared to be superior to human experts, primarily because they can consider a much larger set of possible solutions (several orders of magnitude larger). They do not miss unlikely or unexpected possibilities, as long as these possibilities have been captured within the knowledge base of the system.

Preservation of expertise is also an important aspect of KBSs. The fact that the knowledge of an expert has been captured and codified in a formal fashion provides an archive of information that may be used when the expert is not available. This is vital to the space industry in which knowledge capture has the potential to play a significant role in the evolution of future space programs.

The ability to distribute expertise that is typically localized is another significant advantage of KBSs. They are a means of providing expertise to people who may be acquainted with a particular problem domain but may not be thoroughly proficient at problem solving within that domain. With the aid of expert systems, both the crew and ground controllers can maintain the health of Space Station Freedom without being required to possess an overwhelming amount of knowledge.

KBSs have two major components: a *knowledge base* and an *inference engine* (see Figure A-1). It is the knowledge base that contains a codified version of the expert's knowledge. Furthermore, it is the knowledge base that contains general knowledge about the domain that may or may not be considered part of the expertise. This is why many people call programs that incorporate such information KBSs rather than expert systems. The knowledge base contains two types of information: *declarative knowledge* and *procedural knowledge*. Declarative knowledge consists of the facts that describe the objects of the domain and possible operating characteristics of the objects if applicable to the domain. Procedural knowledge consists of information about the action to be taken in certain situations based on the current state of the declarative knowledge. Both types of knowledge can take on many different forms; such forms are rules, objects/frames, and models, depending on the type of inference mechanism used.

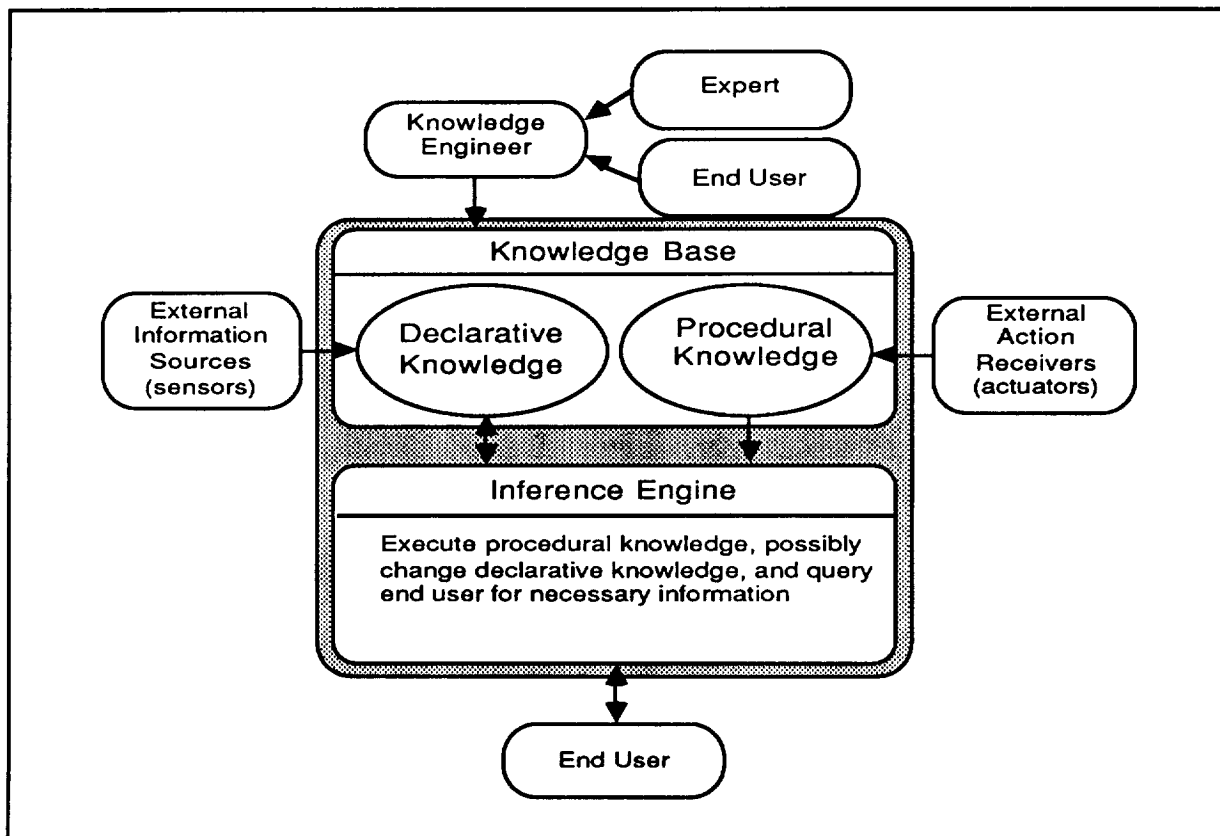


Figure A-1. General Architecture of Knowledge-Based Systems

The inference engine interprets the knowledge base. It embodies the strategies that are used to decide the order in which knowledge is processed, how to find missing information, and how to determine when an acceptable solution has been reached.

The fact that the inference engine and the knowledge base are separate entities allows the use of a single inference engine to process knowledge bases from different domains. Also, a single knowledge base may be processed by different types of inference engines. Thus expert system development environments (or shells) support the acquisition of knowledge from a knowledge engineer, who extracts the expert's knowledge and encodes it in the knowledge representation appropriate for the problem and the

chosen inference engine. The inference engine is responsible for the execution of the procedural knowledge which may result in changing declarative knowledge directly by the conclusions made, or indirectly by the queries made of the user. In addition, many KBSs allow declarative knowledge to be changed by external information sources such as sensors and allow procedural knowledge to instigate external actions such as commands to actuators.

KBSs differ significantly according to the inference mechanism and the knowledge representation used. The following sections introduce the architectures, inference mechanisms, and knowledge representation schemas used within KBSs. Table A-2 lists these topics, along with other advanced automation topics introduced below.

Table A-2. Advanced Automation Architectures, Techniques, and Interfaces

Advanced Automation Architectures	
Logic-based Architectures Rule-based Architectures Frame-based Architectures Neural Network-based Architectures	Semantic Net Architectures Blackboard Architectures Model-based Architectures Fuzzy Logic-based Architectures
Inferencing Techniques	
State-space Search Generate and Test	Forward/Backward Chaining Discrete Event Simulation
Knowledge Representation Schemas	
Logic Clauses Semantic Networks Neural Networks	Rules Frames/Objects
AI Man-Machine Interface Extensions	
Natural Language Understanding Speech Synthesis Other Human-like Perceptions	Continuous Speech Recognition Vision and Image Processing

A.2.1.1 Logic-Based Systems

Logic-based systems were one of the first attempts made by AI researchers to codify knowledge and to infer or reason upon that knowledge. They were influenced by formal logic, the classical approach to representing knowledge and making inferences from facts. Formal logic was developed long before the advent of modern electronic computers and used by philosophers and mathematicians as a logic based calculus (i.e., predicate calculus).

The primary aspect of logic-based systems is their ability to derive new facts based on given or known set of facts. For example, assume it is known that "All normal dogs have four legs" and that "Fido is a normal dog," then it can be inferred that "Fido has four legs."

Prolog is a notable example of a logic based language in which both the facts and derivations of additional facts may be represented in the form of clauses.

A.2.1.2 Semantic Net Systems

Semantic nets use a knowledge representation schema that allows the declarative knowledge to be represented *semantically*; thus, supporting the translation to many different forms of *syntax*. Information is linked together with descriptive links such as: is-part-of, is-a, sub-class-of. The key feature is that important associations may be made explicitly rather than having to be derived. Relevant facts about an object or concept can be inferred directly from the connection of the nodes in the net, as opposed to searching a large data base.

A.2.1.3 Rule-based/Production Systems

Rule-based systems, often called production systems, contain procedural knowledge represented as *rules* typically in the following form:

IF <premises> THEN <assertions>

These rules are the heuristics of the expert system and may be interpreted in various fashions. For example, if the premises within the left-hand side of a rule are all satisfied (true or false) by facts within the knowledge base, the inference engine will execute the assertions on the right-hand side of the rule. These assertions may create or alter facts within the knowledge base or they may execute procedural code.

A.2.1.4 Blackboard Systems

Blackboard systems implement an *opportunistic* approach in which pieces of knowledge are applied either backward or forward at the most opportune time (Nii, 1986^a, 1986^b). The goal of blackboard systems is to extend the robustness of rule-based systems to deal with the declarative and procedural knowledge in an ad hoc manner. They are an excellent approach to integrating expertise from multiple domains. This is especially important when there does not appear to be a common vocabulary for communicating across domains or when the solutions of such domains require significantly different problem solving (inferencing) techniques. This is accomplished by the use of multiple *knowledge sources* and a common communication area referred to as the *blackboard*.

A.2.1.5 Model-Based Systems

Model-based systems are expert systems in which models have been incorporated with rules to provide a robust representation of the structural and operational characteristics of a physical system. Model-based systems embody a level of knowledge typically deeper than that possessed by strictly rule-based systems. This results in an expert system with significantly higher fidelity. Models can be described either quantitatively using conventional algorithms or qualitatively using naive physics encoded with some basic heuristics.

A.2.1.6 Frame-Based Systems

Frame-based systems utilize the expressive power associated with object-oriented programming approaches. Knowledge about an object is stored within a *frame*. Each piece of information within a

frame is stored within a *slot*. Frames and slots are named. For instance, a frame used to describe a car might have a slot called "Number of Wheels"; for most cars this slot would then contain the number 4. An important aspect of frame-based systems is inheritance. Frames are excellent for storing information about classes of objects in a hierarchical or tree-like fashion. Inheritance allows objects which are lower in the tree to automatically inherit characteristics of objects which are higher in the tree. For example, a frame describing a Corvette would not have to explicitly denote that a Corvette has 4 wheels since it can inherit this information from the frame describing cars. If a model was described that had only 3 wheels, then this information would be explicitly declared and would override any inheritance. Slots can also store executable code. These procedures can be invoked upon the occurrence of certain events such as a change to a specific slot's value or the specific value within a slot. Therefore, the inference mechanism of frame-based systems is contained within these procedures.

A.2.1.7 Hybrid Systems

Hybrid systems are potentially the most viable manifestation of expert system technologies. None of the above expert system architectures can claim to solve all problems; in fact, most are suitable for solving only limited classes of problems. Hybrid systems are the cooperative combination of any of the above expert system methodologies; possibly with conventional procedural languages as well. Space Station Freedom will ultimately include hybrid systems that incorporate the heuristic nature of rule-based systems, the qualitative/quantitative models of model-based systems, and the object-oriented knowledge representation of frame-based systems, with separable applications tied together by the cooperative opportunistic aspect of blackboard architectures. Furthermore, conventional procedures will strongly influence the decision making processes by providing additional information and they will also perform routine tasks.

A.2.2 Emerging Advanced Automation Technologies

Recently, other forms of AI research related to automated decision making are also maturing. These include artificial neural networks and fuzzy logic systems. These emerging advanced technologies, and others yet to come, may be additional or replacement components of the hybrid systems discussed above. For example, conventional image processing techniques could be replaced by the pattern recognition power of neural networks, and conventional algorithms for dealing with uncertainty may be replaced with fuzzy logic.

A.2.2.1 Artificial Neural Systems

In the past twenty years, studies of the function and structure of the mammalian brain, and innovative techniques developed by computer scientists, have allowed the development of a powerful new computing paradigm known as Artificial Neural Networks (Lippmann, 1987) [also known as Parallel Distributed Processing (Rumelhart and McClelland, 1986), and Neural Computing (Anderson and Rosenfeld, 1988)]. This paradigm is mathematically and logically based, and is neurophysiologically inspired.

Neural networks have been applied to tasks which humans find easy (e.g., vision (Oyster et al., 1986; Koch et al., 1986; Golden, 1986), movement (Jorgensen, 1987), pattern recognition (Sayeh and Han, 1987; Fukushima, 1986), image analysis (Cottrell et al., 1986; Oyster and Skrzypek, 1987), speech (Sejnowski and Rosenberg, 1986; Elman and Zipser, 1987), but are not very useful for the types of problems which algorithmic computing is often used (number crunching). Because neural networks are excellent pattern matching and classification systems, and because the parallel nature of neural networks allows them to operate very rapidly, they are highly suited for many space-borne applications.

Neural network models are characterized by multiple, non-linear computing nodes (analogous to biological neurons) which are highly interconnected in layers and operate in a parallel fashion

A.2.2.2 Fuzzy Logic-Based Systems

Human beings have a remarkable ability to make decisions or give approximate answers to questions based on knowledge that is inexact, incomplete, or not totally reliable. Fuzzy logic, based on fuzzy set theory, was developed as a method to allow computers to deal with these types of problems.

Fuzzy set theory was introduced by Lofti Zadeh (Zadeh, 1965) as a generalization of classical set theory. A fuzzy set does not have the sharp boundaries of a nonfuzzy set. An object is either a member or a nonmember of a nonfuzzy set, but it has a *degree (or grade) of membership* (ranging from 0 to 1) in a fuzzy set. Fuzzy sets are often derived from qualitative natural language concepts such as *small*, *medium*, and *large*. A graph of the degree of membership of the numbers from 1 to 100 in the fuzzy sets *small numbers*, *medium numbers*, and *large numbers* is given in Figure A-3. Fuzzy set membership is of course application specific, but the algorithms dealing with fuzzy sets are independent of the application and mathematically rigorous.

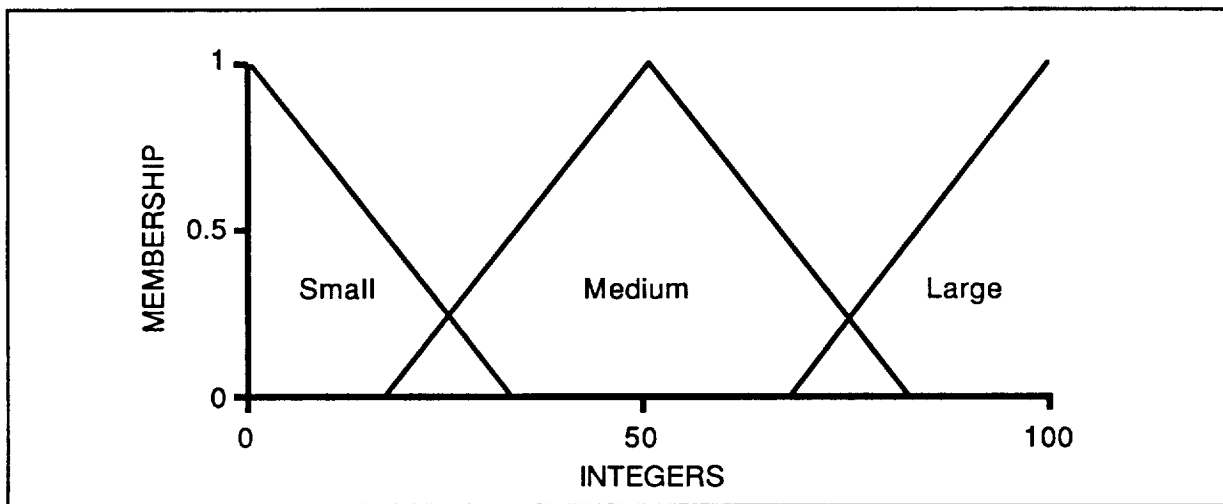


Figure A-3. Fuzzy Set Membership: *Small*, *Medium*, and *Large* Integers

Fuzzy logic is an extension of classical two-valued and multi-valued logic. In two-valued logic a statement is either true or false. In classical multi-valued logic a statement can have a truth value between 0 (false) and 1 (true), where intermediate values represent degrees of truth. Statements in fuzzy logic have truth values which are themselves fuzzy (very true, mostly true, not quite true, unlikely, etc.). Fuzzy logic can also deal with inexact concepts (small, large, near, far), imprecise quantifiers (few, several, many), and modifiers such as *slightly* or *much* (Zadeh, 1988).

Areas of application of fuzzy sets and logic include expert systems (Zadeh, 1983), pattern classification (Kandel, 1982), knowledge representation (Zadeh, October 1983), and control theory (Majers and Sherif, 1985; Mamdani and B. R. Gaines, 1981). Expert system shells incorporating fuzzy logic are commercially available. Currently fuzzy logic is implemented in software. However, a fuzzy logic chip was developed by Bell Labs in 1985 and a fuzzy computer has been built at Kumamoto University in Japan. This computer is capable of performing 10 megaFLIPS (fuzzy logical inferences per second).

(analogous to biological neural networks). An example framework for a neural network is shown in Figure A-2. Nodes within a neural network are arranged into layers and are connected by weights which are adaptive during training. Training of neural networks can take place in a supervised or unsupervised manner.

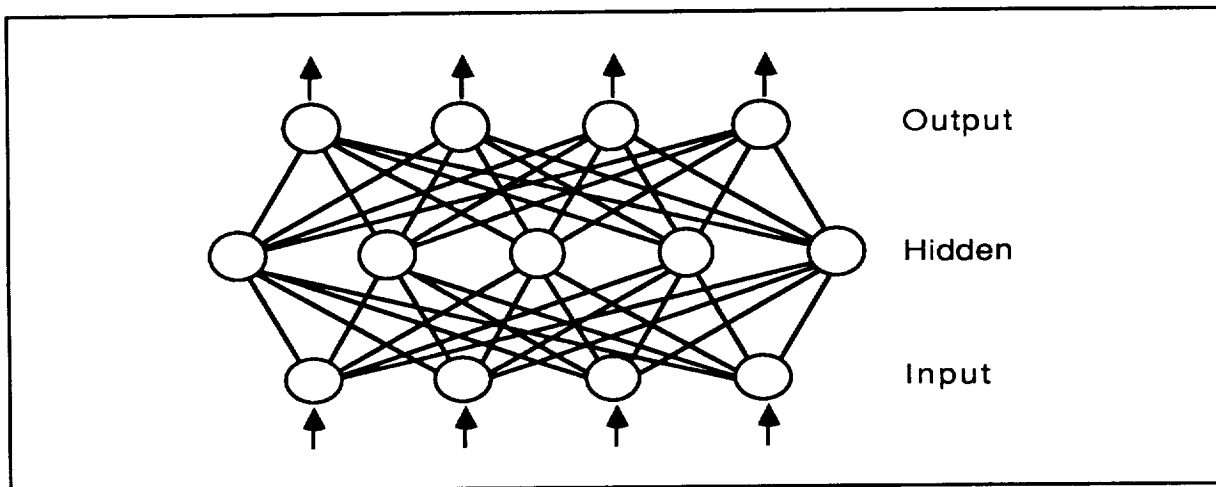


Figure A-2. Example Neural Network Model

In supervised learning, input/output training pairs are used to train the network. Initially, the weights of the network are randomly generated. Stimuli (e.g., some input pattern) are introduced to the input layer. This activation of each input node is multiplied by the weights which connect that node to the hidden layer(s) nodes. Each hidden layer node sums these inputs (both excitatory and inhibitory) and this sum is fed to an activation function which determines whether the hidden layer node will *fire*, i.e., what the output of that node should be.

In a like manner, the hidden layer outputs are multiplied by the weights connecting the hidden and output layers to compute the activation (output) of each output layer node. This observed output pattern is compared to the expected output pattern (associated with this particular pair of input/output patterns) and the difference is used to adjust the weights of the network to allow it to recognize this specific input pattern and produce the correct output pattern. This procedure is repeated for all input/output pairs until the network can correctly identify all patterns correctly.

In unsupervised learning, input patterns are used, but not output patterns. The network simply associates similar input patterns into the same category. There are many different neural network models which use different learning algorithms to adjust the weights during training, each being more suited to particular tasks.

Once a neural network has learned to correctly associate input patterns to output patterns, or output categories, the network can be used to classify or identify patterns introduced to its input layer nodes. Using massively-parallel hardware, the time required to generate an answer will not depend on the size of the network; however, most applications are currently implemented using conventional software simulations and, in most cases, solutions are produced much more rapidly than typical rule-based expert systems. If the network is implemented on truly parallel hardware, then it will most likely out-perform many conventional algorithms running on serial machines.

A.2.3 Advanced Automation Interfaces

Significant advances in human-computer interfaces have been made due to AI research. High-resolution bitmapped displays with window interfaces and graphical input devices are spinoffs from AI research that have already become a mainstay in personal and professional computing. These interfaces support communication between man and computer giving more freedom to the end user interact in an ad hoc manner. Although these interfaces still require that the user learn certain skills, they make interaction with the computer more user friendly.

AI research has long been concerned with making the interaction between the human and the computer closer to the manner in which humans interact with each other. Natural language, speech recognition, speech synthesis, and vision and image processing are examples of AI computer interfaces.

A.2.3.1 Natural Language Understanding

The goal of natural language understanding is to allow a user of an application to enter commands or data to the computer using the complete, natural language (e.g., English or Japanese). In other words, the computer must understand the semantics of this language to allow it to communicate with the user in a manner constrained only by that language. According to Barr and Feigenbaum (1981),

So far, programs have been written that are quite successful at processing somewhat constrained input. The user is limited in either the structural variation of his sentences (syntax constrained by an artificial *grammar*) or in the number of things he can mean (in domains with constrained semantics).

Although this area of AI has progressed to a degree adequate enough to provide sophisticated front ends for many applications, it still has a long way to go before computers can understand a full range of meaning in a language such as English.

The most notable examples of natural language understanding have been applied to domains involving robot control and data base query. These domains are highly suited to the current limitations of natural language understanding, since the robot is usually constrained by a relatively small world consisting of predefined objects and a limited set of possible movements and the data base is constrained by the types of queries that may be performed and a limited number of fields that may be accessed.

The primary disadvantage of comprehensive natural language understanding is that it may embody too many human characteristics. That is, humans are very prone to making unclear or ambiguous statements and are susceptible to misunderstanding others. In space borne applications, there is usually little room for such an error. There are advantages to having constrained vocabularies when operation or control are concerned. This is illustrated by the command language used in any branch of the military. The military attaches precise meaning to certain commands; thus, guaranteeing that when an order is given, the commanding officer can be assured that he is understood completely.

A.2.3.2 Continuous Speech Recognition

Even the modest natural language understanding systems available today constrain the user by allowing only typed input. Continuous speech recognition is the mechanism by which the user may be relieved of the hands-on duty normally required to interact with a computer. Recognition of connected speech, combined with natural language processing, will allow an astronaut to tell a computer to do specific tasks in addition to answering questions posed by the computer.

Speech recognition systems exist today, however, these systems are very limited and are usually constrained to recognition of single words spoken alone and slowly. These systems must also be trained

by each user to recognize the words spoken by a particular user. Continuous speech recognition, on the other hand, is a much more difficult problem. Problems arise from the need to separate one spoken word from the next or previously spoken word. In addition, speech must be separated from background noise such as other conversations and the sound of operating machines. The computer must also be able to realize when the user is speaking to it and not to someone else in the same vicinity. For instance, if the user says, "If I say 'Start the engine', the computer will fire the main engine," the computer must realize that the user is speaking to someone else and not pick out the words "start the engine" from the sentence and execute the command.

A.2.3.3 Speech Synthesis

Providing a computer the capability to communicate with a user in the user's natural language will be essential in many situations. For example, an astronaut's attention can be drawn by auditory warnings and suggestions more easily than by conventional messages presented on a display device simply because the astronaut does not have to continually look at something in order to receive the message. Speech synthesis devices exist today but many of them are constrained to the point where they sound robot-like.

A.2.3.4 Vision and Image Processing

Vision and image processing is currently being utilized in many factories and laboratories to perform a variety of tasks. It has several sub-fields that range from simple picture processing (noise filtering) and object classification to complex scene analysis. Vision and image processing will enhance the autonomous ability of the intelligent control systems of Space Station Freedom. Furthermore, with active vision systems the computer may actually watch the astronaut and be able to predict his or her interest based on eye movement and any other gestures.

A.3 CANDIDATE APPLICATIONS FOR SPACE STATION FREEDOM

The Space Station Freedom Program (SSFP) can incorporate advanced automation in both ground and onboard software systems. Hundreds of potential applications for use onboard Space Station Freedom have been identified in SSFP documentation and in studies. These lists can be reduced to several general application areas common to many Space Station Freedom applications. For example, Boeing (November, 1987) extracted a list of 60 candidates for a group they called *autonomous system managers*. This category of application was defined to embody sensing, data collection and analysis, trend development, automatic planning, automatic decision making and interaction with human crew members. A subset of this category (37 candidates) were applications with a smart analyzer function. These included such applications as performance monitors, medical monitors, automatic calibrators, and fault predictor/analyzer/managers. All of the candidates in the smart analyzer group have similar characteristics that include collection of sensor data, comparison of that data with data bases or models, automatic inferencing, and the output of results as commands and displays.

The following list is another classification of the basic functional characteristics of advanced automation candidates for Space Station Freedom systems:

- Fault Detection Isolation and Recovery (FDIR)
- Control and Monitoring
- Caution and Warning
- Intelligent Reconfiguration

- Planning and Resource Management

A.4 ADVANCED AUTOMATION TECHNOLOGY READINESS

The readiness of the various advanced automation technologies, tools, and application areas will play a large role in our discussions of evolution paths for advanced automation in Section 4. The following paragraphs, figures, and tables serve to introduce concepts related to technology readiness *by example*, using technology assessment curves and candidate application technology readiness information generated in a preceding study. It is not our goal here to explicitly identify the technology readiness level of advanced automation technologies, tools, or candidate applications. This information was taken from Boeing (1987) and the authors do not necessarily agree with or defend the accuracy of this information. They simply suggest that technology readiness information will need to be derived relative to the various applications, and the underlying advanced automation and conventional technologies, used within the SSFP for effective planning and evolutionary growth.

A.4.1 Technology Assessment Curves

The Boeing report identified the following technologies which would be required for both the autonomous system manager and the fault predictor/analyzer/manager systems: computing, KBSs, planning, and speech. The following figures present technology assessment curves for each of these technologies.

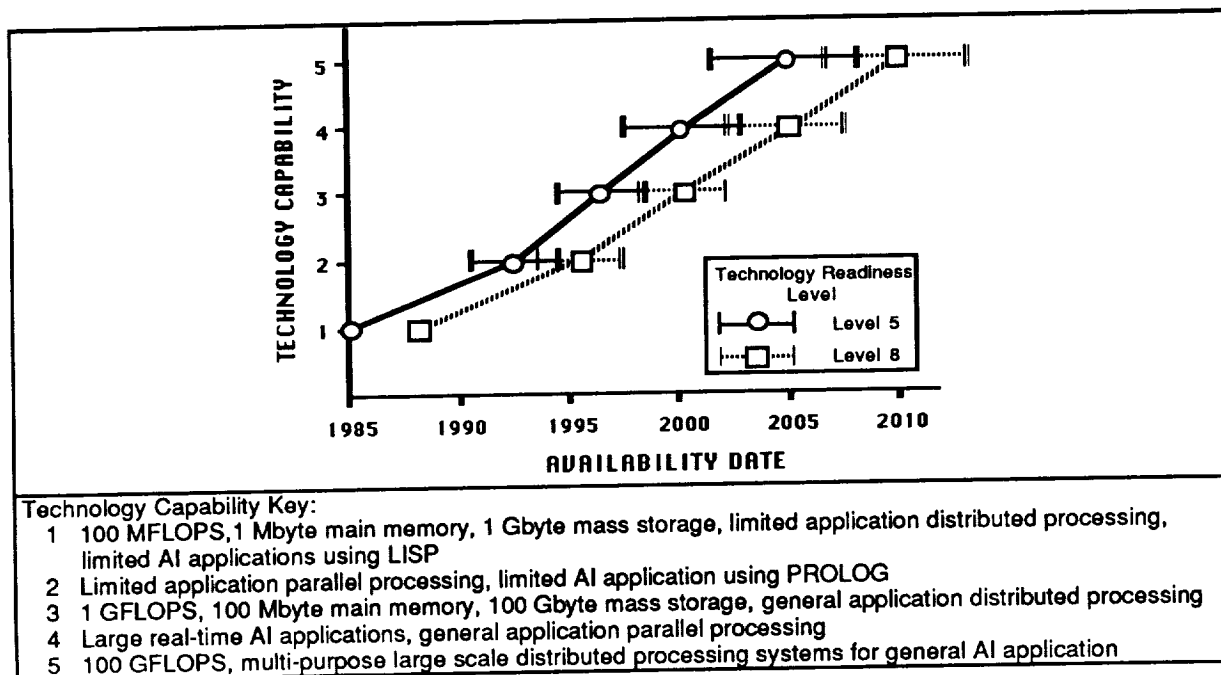


Figure A-4. Computing Technology Assessment Curve

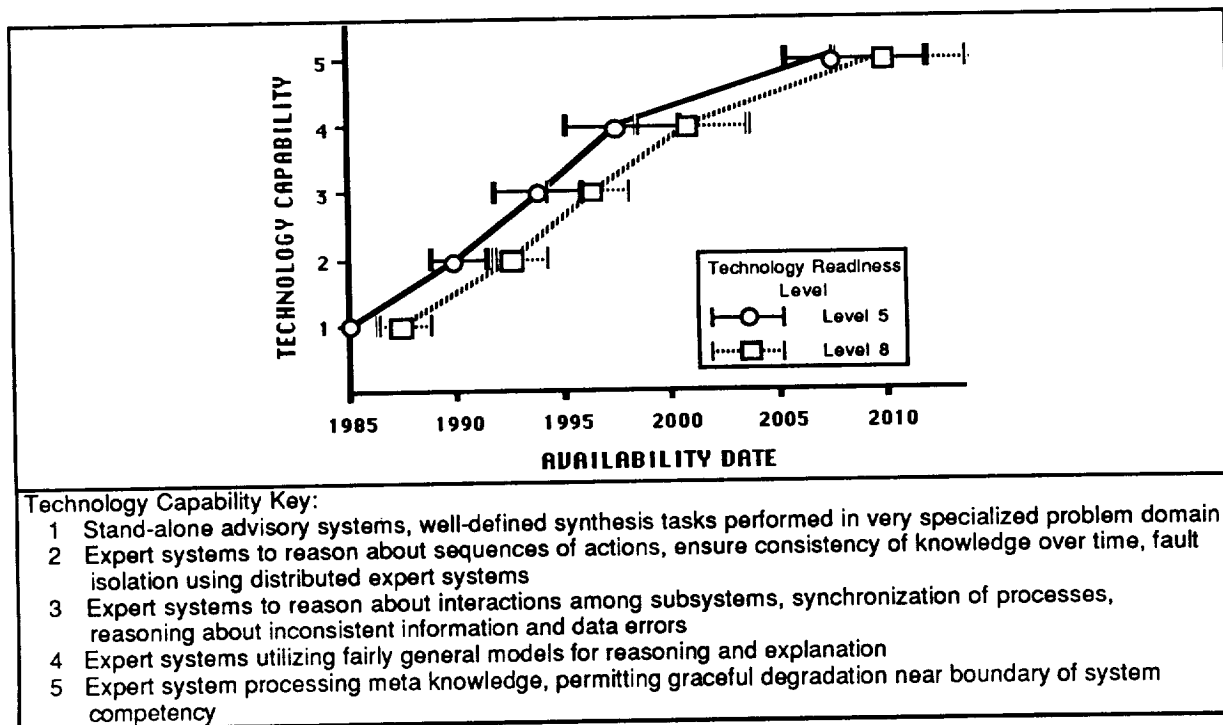


Figure A-5. KBS Technology Assessment Curve

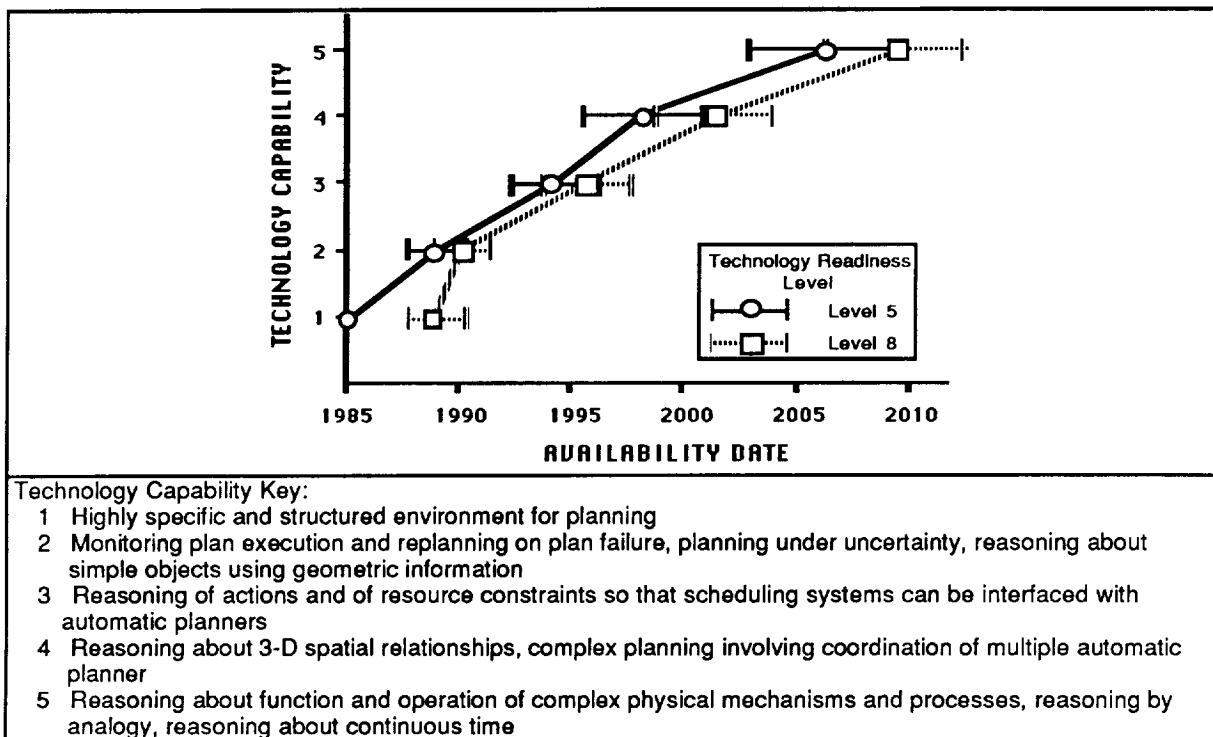


Figure A-6. Planning Technology Assessment Curve

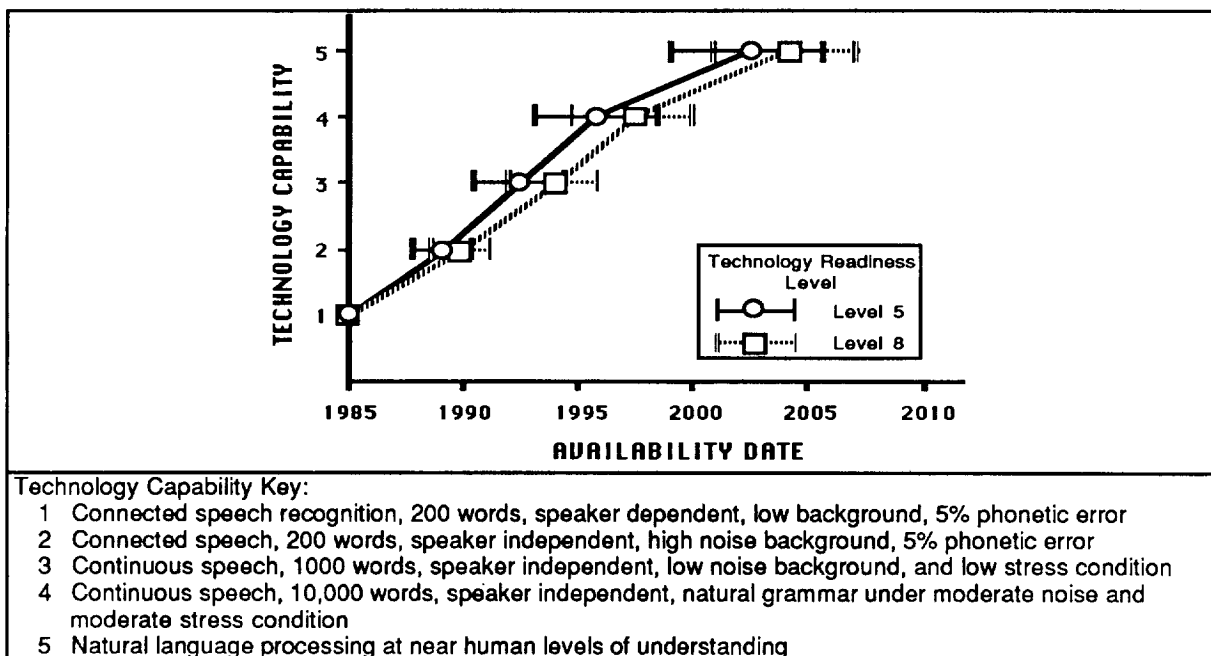


Figure A-7. Speech Technology Assessment Curve

A.4.2 Candidate Application Technology Readiness

The following tables present information from the Boeing report relative to the availability dates of level 6 technology readiness (i.e., prototype or engineering model tested in relevant environment). Again, these data serve only as examples.

Table A-3. Autonomous System Manager Level 6 Readiness

Technology	Technology Capability Level		Level 6 Year
Speech	3	Continuous speech, 200 words, speaker independent, low noise background, and low stress condition	1992
Planning	3	Reasoning of actions and of resource constraints so that scheduling systems can be interfaced with automatic planners	1995
Computing	3	1 GFLOPS, 100 Mbyte main memory, 100 Gbyte mass storage, general application distributed processing	1996
KBS	3.5	Beyond the capability of expert systems to reason about interactions among subsystems, synchronization of processes, reasoning about inconsistent information and data errors, but not yet to the stage of utilizing fairly general models for reasoning and explanation	1996

Table A-4. Fault Predictor/Analyzer/Manager Level 6 Readiness

Technology	Technology Capability Level		Level 6 Year
Speech	4	Continuous speech, 10,000 words, speaker independent, natural grammar, under moderate noise and moderate stress condition	1995
Computing	4	Large real-time AI applications, general application parallel processing	1996
Planning	4	Reasoning about 3-D spatial relationships, complex planning involving coordination of multiple planners	1998
KBS	4	Expert systems utilizing fairly general models for reasoning and explanation	1999

APPENDIX B

SOFTWARE ENGINEERING FOR ADVANCED AUTOMATION

There is growing recognition of the importance of integration, portability, fielding, testability, maintainability, debugging support, and documentation in the design of Knowledge-Based Systems (KBSs). Software engineering is now being accepted as applicable to the development of a KBS that is carried through to delivery and maintenance. When software engineering is applied to KBSs, they should be recognized as real, deliverable software that is judged, like conventional software, by its long-term reliability and performance in end-user environments (Rothenberg et al., 1987). This section addresses the software engineering activities relative to KBSs and the software life cycle required to manage these activities.

B.1 SOFTWARE ENGINEERING ACTIVITIES

KBSs are not magic; they are just a new kind of software. There are many similarities between the development process for a KBS and the development process for a decision support system or a simulation model (Friel and Mayer, 1985). The development activities correspond in an approximate way to those performed in a conventional software development project, although they are performed in a somewhat different fashion. The following paragraphs address the software engineering activities and the environment, i.e., the techniques, methods and tools, required to support advanced automation software throughout the life cycle.

B.1.1 Concept Formulation

The development process for a KBS starts with the identification of an appropriate problem. This process begins with an assessment of the fundamental requirements that the system must accomplish. Based on these basic requirements, the task is characterized, the knowledge domain, representation requirements, and world interface requirements are roughly estimated; the reasoning process involved is determined; and finally the decision is made on the admissibility of the task to a knowledge-based solution (Friel and Mayer, 1988). The problem must be one which cannot satisfactorily be solved by conventional computer techniques. It should have symbolic structure and there should be heuristics available for its solution. For a problem to be amenable to solution by a KBS at this time, the problem domain must be narrow and well bounded, the task must require cognitive skills instead of common sense, and the problem must be clearly defined and understood (Turban, 1988).

B.1.2 Requirements Definition Activities

For a KBS, the first step in defining the requirements is a front-end analysis to provide a high level statement of system functional design requirements (i.e., major goals and functions of the system) to later support the validation of the system. Requirements documentation developed at this stage is usually incomplete. The requirements should specify money, time, and other resources for the prototyping that is frequently required to define more complete requirements.

Rapid prototyping is central to the methodology for defining the detailed requirements for a KBS. A quickly built, small-scale version of the system generates an understanding of the problem and tests the feasibility of the KBS so that the real system can then be specified and implemented. This small-scale working program is used to explore the application domain with the participation of the expert. The prototype helps to determine the structure of the knowledge base; it accelerates the process of knowl-

edge acquisition; and it provides information about the initial definition of the problem domain (Turban, 1988).

At the conclusion of the rapid prototyping, the initial requirements documents are augmented by the developers, domain expert, and the users. Decisions are made about project directions. At this time, it can be decided whether to do another prototype, to discontinue the project because the high level requirements cannot be met, or to move into the iterative development phase of the software life cycle (Richardson and Wong, 1987).

B.1.3 Design Activities

The design of the early KBSs generally was ad hoc, one of a kind, and difficult to manage. As KBSs move out of the lab and into industry, the systems are being designed for testability and for maintainability. The design process for a KBS is more closely related to the design process associated with a simulation model than to the design activity normally associated with other conventional software. The designer attempts to create a problem solving mechanism that emulates the reasoning processes and the knowledge of the human expert. The result of this activity is a design of the control strategy (i.e., the system architecture) and the knowledge representation strategy for the system, followed by the selection of a tool that supports these strategies. Documentation of all design decisions and the constraints that affected these decisions supports testability and maintainability.

B.1.3.1 Architecture Selection

The architecture of a KBS is the design of the control component (i.e., inferencing mechanism) for the KBS. The architecture represents the problem solving strategy for the system. Citrenbaum and Geissman (1986) suggest that the architecture be selected based on the size of the solution space. The choices range from simple exhaustive search to a set of cooperating subsystems following multiple lines of reasoning. The approach to chaining of rules depends on the depth and breadth of the problem, the anticipated strength of inferences, and the extent to which subproblems are likely to interact. It may be data-driven (i.e., forward chaining), goal-driven (i.e., backward chaining), mixed forward and backward chaining, opportunistic (i.e., blackboard control architecture), frame-based with inheritance networks, script-based, or goal-driven logic (i.e., PROLOG). Search may be depth- or breadth-first, or follow some optimizing strategy.

Geissman and Schultz (1988) define *structured design* of a KBS as formally defining one or more inferencing mechanisms, selected from a limited set of inferencing mechanisms that are certified, as compilers are certified for conventional programming languages. Verification and validation (V&V) is independently performed on any escapes to procedural code that might modify working memory or the value of an item in an inheritance network. As candidate inferencing mechanisms for certification, Geissman and Schultz identify frames with inheritance networks, backward- and forward-chaining production systems, and goal-driven logic (i.e., PROLOG), all of which represent discrete event networks such that the future state of the system is the result of the initial state and a number of transitions. Tools have been built to analyze the logical consistency of a knowledge base expressed in terms of one of these inferencing mechanisms.

B.1.3.2 Knowledge Representation

Because a large amount of knowledge is needed for a KBS, a good knowledge representation scheme is required to support the access, update, and maintenance of this knowledge. KBSs use semantic networks, frames, Prolog clauses, and many other schemes for presenting knowledge. Each has its advantages and its weaknesses. However, they all provide a high level of abstraction and separate the problem solver from the knowledge.

Jacob and Froscher (1987) have proposed a design methodology for rule-based systems that divides the set of rules into groups of rules and then concentrates on those facts that carry information between rules in different groups. The knowledge engineer declares groups of rules, flags all between-group facts, and provides descriptions of those facts to any rule groups that use such facts. These descriptions help the programmers in updating or maintaining the knowledge base.

B.1.3.3 Tool Selection

Commercial tools that support the development of KBSs make it possible to develop such a system in an order of magnitude less time than would be required with the use of development languages such as Lisp (Gevarter, 1987). These tools range from simple shells, into which information is inserted to create an expert system directly, to a collection of different paradigms incorporated into a single tool, which may be considered a high order programming language or environment. What these tools have in common is the minimum set of a knowledge representation scheme, an inference or search mechanism, a means of describing a problem, and a way to determine the status of a problem while it is being solved (Citrenbaum et al., 1987). Unlike tools for conventional software development, some portion of the support environment that is provided by a tool used to build a KBS becomes part of the completed KBS.

Tools are designed to help the system developer represent knowledge and get an initial version of a KBS working quickly so that the domain expert can observe its behavior and suggest revisions or modifications. For complex KBSs that stretch the capabilities of existing tools, tailored inference engine(s) and search methods may be worth the time and expense. Nevertheless, it is advisable to do rapid prototyping work with an existing tool and then, once the requirements of the system are well understood, make a careful assessment of a custom-built solution.

The large number of tools now on the market and the pace at which new ideas are being incorporated in tools can make the selection of a tool very difficult, but a properly selected tool may provide a close match to the developer and end-user needs. There is no single *best* tool for *all* knowledge-based applications. Virtually each tool is unique and no tool has a large pool of skilled users. Each tool has its own purposes and strengths and may complement other tools by being used at different times in a project's life cycle. For example, one tool might be used as an initial tool for use in rapid prototyping to better understand the requirements and another tool may be used in the iterative development phase of the software life cycle. This might be caused either by the different requirements of the two phases of the life cycle or by changes in the requirements discovered during the initial prototyping phase.

The task of selecting a tool amounts to finding the best *fit* to the intended use. The identification of the basic category of the task to be performed by the KBS, along with the appropriate architecture and knowledge representation to support this category of task, will ultimately have a strong effect on the selection of an implementation tool (Friel and Mayer, 1985). Tool choices should also be guided by the size of the system to be built, how rapidly a system of the given size and complexity can be built with the tool, the speed of operation of the tool both during development and, particularly, during end use. Other important factors are the degree of satisfaction of both the developer and the end user based on properties such as the match of tool features to the problem's characteristics, the developer and end-user interfaces, and the ease of learning. The system's portability, the computers it will run on, the delivery environment, the system's capability of interfacing with other programs and data bases, and whether the developed system can be readily embedded in a larger system are all important in an evaluation of a tool (Gevarter, 1987).

The trend in commercial tools is toward less expensive, faster, more versatile, and more portable tools that will support development of systems that can directly communicate with existing conventional software (e.g., data bases) and can be embedded into larger systems. They are moving from Lisp machines to more conventional workstations and micro computers (Gevarter, 1987). In response to the

program and data base interface and the embeddability problems, current research is also addressing the addition of data-driven, rule-based expert system techniques to an existing, compiled, procedural language while preserving the features, syntax, semantics, and development process of the procedural language (Milliken et al., 1988; Harston and Martinez, 1988).

B.1.4 Development Activities

The objective of the system development activities is to develop and implement the system architecture and knowledge representation using the selected tool. The detailed knowledge about the domain is extracted and encoded. The result is a functioning KBS. These systems require new techniques, referred to as *knowledge engineering*, to extract knowledge from human experts. Although textbooks, handbooks, and data bases are sources of knowledge, human experts provide undocumented knowledge that is more complex than the knowledge found in these documented sources. Human knowledge, based on experience and common sense, often can be expressed in terms of heuristics (i.e., rules of thumb). Such knowledge is hard to extract from people because they cannot articulate just how they solve a problem without applying the knowledge in context.

B.1.4.1 Code

Coding of a KBS is not the final mechanistic step at the end of the design process (as it is in the development of conventional software), but is part of the iterative design process. The system's heuristics and knowledge base grow through an evolutionary process to deal with the full complexity of the problem. Interfaces to other systems, data bases, and data-capture devices, in addition to a sophisticated user interface are developed.

Domain experts and end users are repeatedly shown a working system as it is being developed so that they can test the system with real problems that they have handled. The knowledge and rule base are enhanced based on this exercise, thereby iteratively developing the system. A test case library should be developed in parallel with the coding (Citrenbaum and Geissman, 1986).

B.1.4.2 Unit Test on Development System

Because the testing of a KBS includes determining the correctness of the reasoning, testing is different from that of conventional systems. This presents some new problems for testing. The first problem is that KBSs often show non-deterministic behavior because the conflict resolution strategy can depend on execution-time parameters, making the behavior unrepeatable and therefore more difficult to test. The second problem is that conventional input-output analysis for testing is difficult to use because there is no precise input-output relationship for rules as there are for procedures in conventional software. The third problem is that the number of ways rules can be activated is usually too large to use branch-and-path-coverage tools (Ramamoorthy et al., 1987).

One answer to these problems is to have users, in addition to the domain expert, evaluate the system as it is being iteratively developed. This evaluation is done in a simulated environment where the system is exposed to test problems, either historical cases or cases provided by the users. When this evaluation reveals cases not handled by the system, new rules are added or old rules are modified. Each time a substantial change is made, testing must be repeated. When the system reaches an acceptable level of stability and quality, it may be ported to a delivery system prior to integrated testing.

B.1.4.3 Port to Delivery System

Small systems are being developed, implemented, and used on PCs today. However, porting to a delivery system is usually required after large systems are developed. In order to take advantage of the powerful tools that support iterative development of KBSs, large systems are frequently developed on

special-purpose workstations or on standard computers other than the computer on which the completed system will reside. The system may be required to execute on specified hardware (e.g., Standard Data Processors (SDP) of Space Station Freedom). The optimization of performance may require a delivery system different from the development system. Usually, in order to improve performance, only sub-elements of the tool (with the development portion removed) act as a software delivery vehicle for the developed expert system. Alternatively, the whole system may be rewritten in an approved or faster, more portable software set, such as Ada, C, or CLIPS (a NASA developed shell).

B.1.5 Unit Test on Delivery System

The battery of tests developed for unit testing on the development system are repeated on the delivery system in a simulated environment. This is followed by integrated testing to check out the system's interfaces.

B.1.6 Independent Verification/Acceptance Activities

V&V should be included as an important activity within each phase of the software life cycle. New V&V methods will be required for each technology only where there are new characteristics which cannot be tested using existing methods. User acceptance is the final measure of success of a KBS. In many instances, the users will decide whether to use a system and how to use it. Users must evaluate for themselves how effectively they are able to use the system in their real world domain (Richardson and Wong, 1987).

B.1.7 Operations Activities

Usually a KBS is used in parallel with a human expert for some period of time. Many systems never reach finalization; they are continually being developed. Nevertheless, documentation and maintenance plans are just as important as for conventional software (Turban, 1988). Many tools used to support the operations of conventional programs can be adapted for KBS programming. These tools include version management systems, configuration management systems, and modification request systems (Ramamoorthy et al., 1987).

B.1.7.1 Configuration Management

At some time in the development, the on-going enhancement of the KBS must be separated from the maintenance of the system. Two copies are made of the system. One copy is *frozen* and is maintained but no longer enhanced in any significant way. Enhancement continues on the other copy, which will continue to evolve and eventually replace the frozen system. Tools and methodologies to support this configuration management activity are like those for conventional software.

B.1.7.2 Maintenance

The objective of maintenance is to evaluate and enhance the system as necessary to improve system performance. In a conventional system, maintenance occurs only in the operations stage and refers to the correction of bugs and changes to the code. Maintenance and modification of a KBS is an important and continuous part of the development as well as the operational activities because the emphasis at all stages is on the growth of the knowledge base. Rules evolve with the experience of their use and are therefore modified more often than algorithms. Because rules can be time-dependent, their validity can change over time (Ramamoorthy et al., 1987).

B.2 SOFTWARE LIFE CYCLE MODELS

Effective management techniques are required for the development and maintenance of any type of software. A number of software life cycle models have evolved to manage the software process for conventional applications. Each model identifies the phases in the development and evolution of software, defines the order of these phases, and establishes the transition criteria for progressing from one phase to the next.

In the earliest days of software development the life cycle model was a two step code-and-fix model. This most often led to poorly structured code that was a poor match to users' needs and was difficult to test and modify. From this experience the needs for requirements, design, test, and maintenance were recognized. A stage-wise model emerged which identified eight sequential stages. This stage-wise model evolved into the well known waterfall model in 1970. More recently, the iterative development, the spiral, and the transform models have been formulated in response to problems in applying the waterfall model to some types of software applications. Richardson and Wong (1987) have identified three important aspects of adaptation of the traditional life cycles to the development of KBSs as changes and improvements to requirements documents, addition of an iterative prototyping loop, and inclusion of the user in the development process.

In the following paragraphs, four life cycle models for conventional software are defined with comments from the literature on the applicability of each as a life cycle model for advanced automation software.

B.2.1 Waterfall Model

The waterfall model is a refinement of the stage-wise model by incorporating successive feedback loops between stages (phases) and confining iterations as much as possible to neighboring steps, to minimize the much more expensive long-range feedback loops. Each phase in the life cycle ends with a verification or validation activity in order to minimize the number of undetected errors. Figure B-1 illustrates the eight steps in the waterfall model. Each phase in the life cycle feeds back information to earlier phases, for refinement of documentation or to provide for regressive iteration. The specific activity at each stage takes place and is completed for the entire system before progression to the next stage.

The waterfall model has become the basis for most software acquisition standards today. This model is a linear progression of phases that are expected to yield correct results because it is based on complete and rigid documents (system specification, software requirements specification, top-level design document, detailed design document) that serve as a basis for verification and validation. The waterfall model is primarily a document-driven approach with documents as completion criteria for early requirements and design phases (Boehm, 1988).

Because a formal and complete specification of the problem is a necessary prerequisite of the waterfall life cycle, standard software engineering methodologies based on rigorous requirements definition have seldom been applied to the development of KBSs. The specifications for a KBS are rarely complete. The exception to this generalization is the case where most of the knowledge is acquired from documented sources.

One example where traditional methodologies were successfully adapted and applied is the KBS developed by Bochsler and Goodwin (1986^a) to demonstrate automated rendezvous and proximity operations onboard a space vehicle during flight. This was possible because documented knowledge sources were readily available. Applicable documentation from the existing Orbital Operations Simulator was utilized for an initial rapid prototype. For Space Station Freedom as the rendezvous target, the rule base was derived from proposed guidelines for Space Station Freedom's command and control zone operations. For a hypothetical orbital maneuvering vehicle as the rendezvous target, the rendezvous profile from the

Shuttle Program was adapted to the hypothetical vehicle. Documented Shuttle flight rules and other relevant documents that were identified during meetings with the domain experts (the trajectory mission planners) provided additional documented knowledge sources.

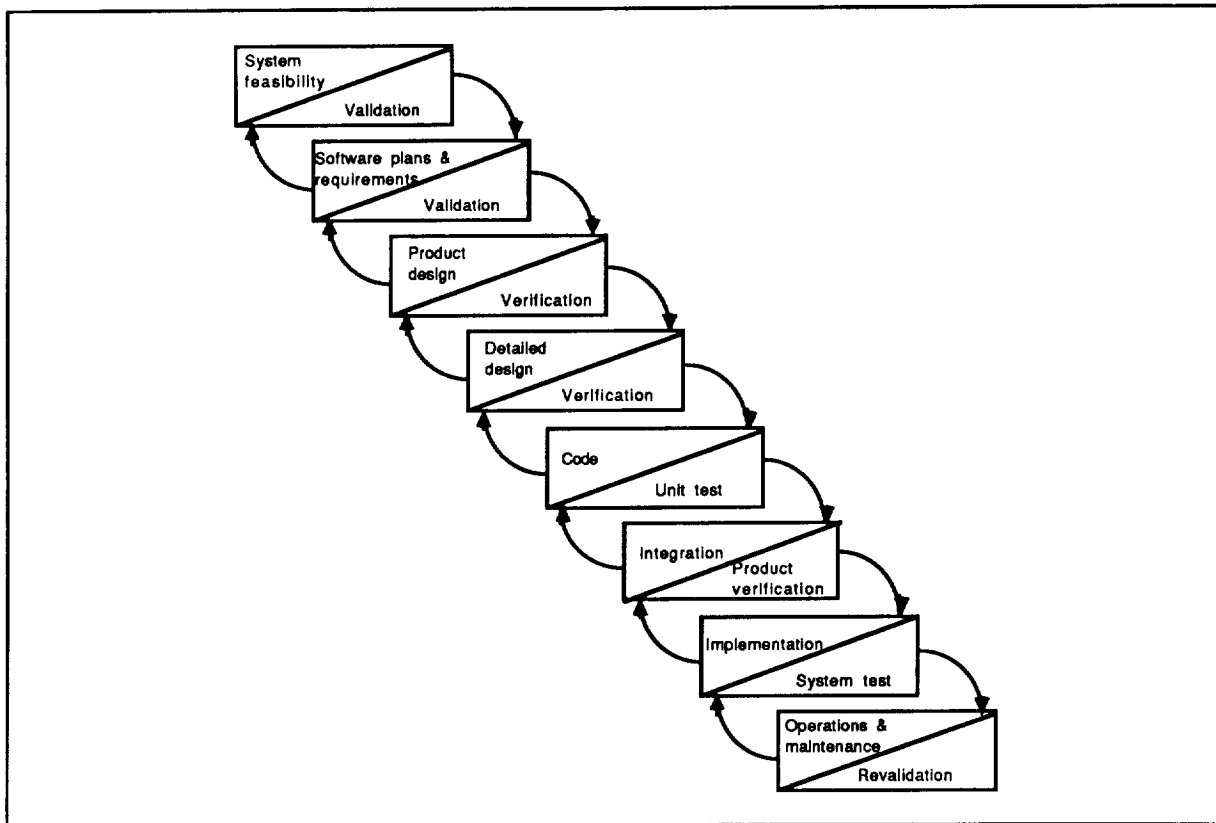


Figure B-1. The Waterfall Model of the Software Life Cycle
(Source: Boehm, 1988)

Formal guidelines and requirements were the products of the requirements phase. A traditional structured, top-down analysis and design technique was used to develop the requirements, construct the design, and support implementation. This approach was augmented by judicious prototyping. A rapid prototype provided insight to the proposed architecture during the definition phase and enhanced communication with the experts while developing detailed system designs. Limited scale prototyping of specific troublesome problems was again used in the design phase to augment the top-down design methodology (Bochsler and Goodwin, 1986^a and 1986^b).

B.2.2. Iterative Development Model

The waterfall model, with its detailed requirements and specifications documentation, does not work well for interactive end-user applications where user interfaces and decision-support functions are often ill-defined. In response to the needs of this type of system, the iterative development model, also referred to as the *evolutionary development model*, has been formulated with improved and more frequent communications between customer and developer (McCracken and Jackson, 1982). This model emphasizes the use of prototyping capabilities to converge on high-leverage software product features

essential to the user's mission. Application users are given a physical representation of key system features long in advance of final implementation. The phases of this model consist of expanding increments of an operational software product with the direction of evolution being determined by operational experience and user acceptance. Boehm (1988) characterizes the iterative development model as a code-driven approach that may lead to poorly modularized code, which is difficult to interface with other independently evolved applications.

Some aspects of KBSs are similar to the characteristics of conventional interactive end-user applications where user interfaces and decision-support functions are often ill-defined at the outset. Therefore, it is not surprising that the most frequently used life cycle model for the development of KBSs has been a variation of some sort on the iterative development model. Two variations on this iterative development theme are presented in the following paragraphs.

The Abacus Programming Corporation has evolved a methodology based on an iterative development life cycle for KBSs. It consists of the following four phases (Geissman and Schultz, 1988):

- 1 Problem Determination (analogous to the conventional requirements definition phase)
- 2 Initial (Rapid) Prototype (analogous to the conventional design phase)
- 3 Iterative Growth (analogous to the conventional coding phase)
- 4 Delivery System (porting to the operational delivery environment and final V&V testing in the delivery environment)

This example of an iterative development life cycle model for KBSs has been strengthened by Culbert et al. (1988) with the addition of some organizational aspects to the life cycle model. They have proposed the addition of a panel approach to provide verification and documentation at the end of each of the four phases identified above. Regular panel review provides verification of both purpose and design of the KBS and requirements specifications. The panel is made up of the major agents involved with a KBS: developers, domain experts, system users, and managers with system responsibility.

Culbert et al. (1988) have further proposed that, at the end of the problem determination phase, an initial requirement review (IRR) be held by the full panel followed by a preliminary design review (PDR) at the end of the initial prototype phase, when the prototype is demonstrated to the panel. Architectural issues and overall feasibility are evaluated and requirements that were generated during this phase are documented. At the end of the iterative growth phase a formal system design review (SDR) is held with the full panel. Requirements are consolidated into a formal System Requirements Document and a test plan, describing how the completed KBS will be verified, is published by the panel. Problems encountered in the testing of the delivery system are reviewed by the panel and the system may be sent back to the iterative growth phase. After it passes testing, the KBS can be put under standard configuration control. Maintenance can be handled in a standard manner, but extensive changes may require further prototyping work and review by the panel.

B.2.3 Spiral Model

The spiral model is a flexible, risk-driven model. It consists of a spiral with four major steps repeated in the same sequence in each cycle at increasing levels of elaboration, as shown in Figure B-2. The major steps within each cycle include (1) identification of objectives, constraints, and alternative means of implementation for this cycle of the product; (2) evaluation of alternatives with identification and resolution of risks; (3) development and verification of next-level product; (4) planning for and commitment to the next cycle.

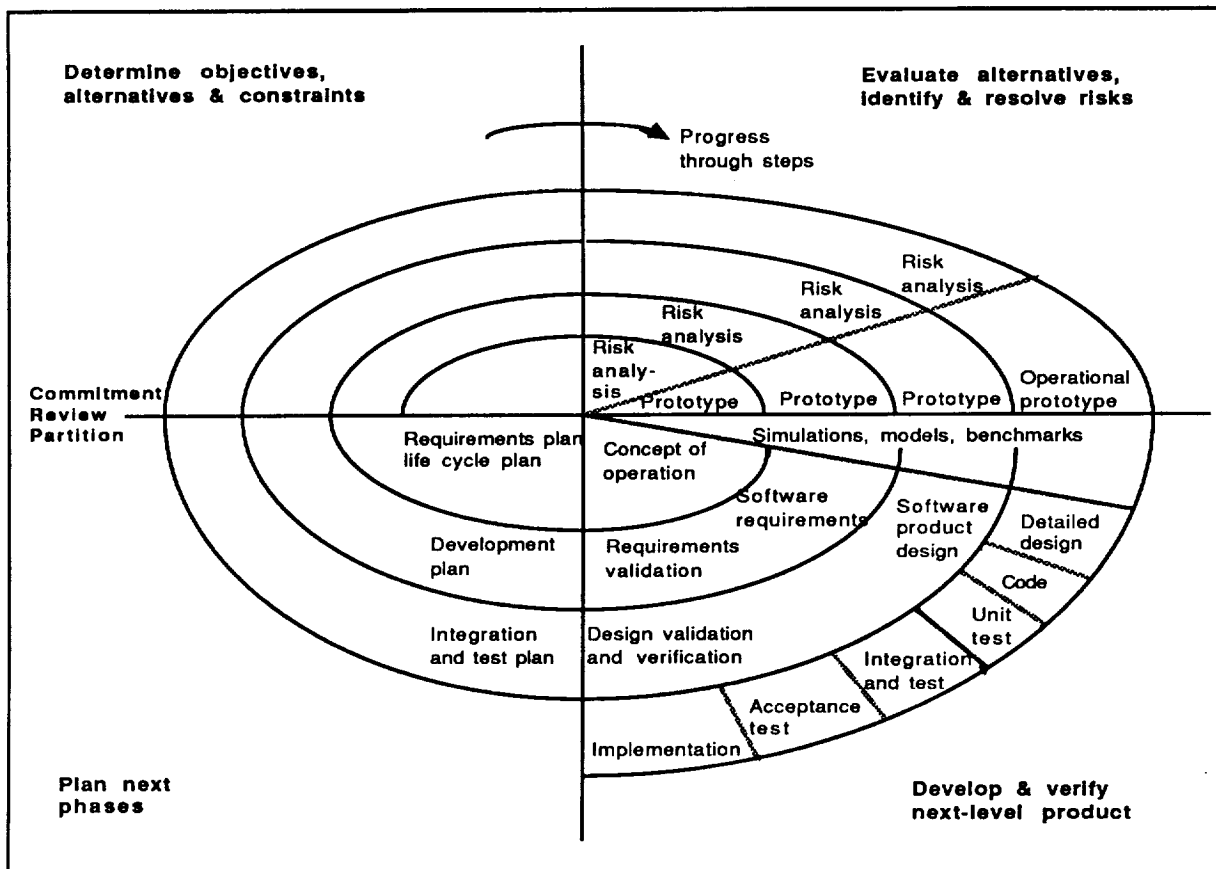


Figure B-2. Spiral Model of Software Life Cycle
(Source: Boehm, 1988)

The major difference between the waterfall model and the spiral model is that incorporated in each cycle of the spiral model are considerations of objectives, alternatives and project risks. The risk resolutions drive the development process. Because risk considerations can lead to the implementation of only a subset of all the potential steps in the model, variations of the spiral model can accommodate most previous models as special cases when appropriate. For example, risk analysis of a project with low risk in user interface and performance but with high risk in budget and schedule control would drive the spiral model toward an equivalent of the waterfall model. On the other hand, risk analysis of a project with high risk in either user interface or performance would drive the spiral toward the iterative development model. In addition, risk-driven specifications can have varying degrees of completeness, formality, and granularity, depending on the relative risks of doing too little or too much formal specification. Prototypes may or may not be used; they may be used to derive requirements or to explore difficult design decisions, and then be thrown away; or, alternatively, they may be iteratively refined into a final product. The major benefit of the spiral model is that it is not a single rigid model applied to all software development projects.

At the present time the spiral model is not as fully elaborated as the more established models. Its primary difficulties are that it does not yet match the world of contract software acquisition, it relies heavily on risk-assessment expertise, and it needs further elaboration of spiral model steps to ensure that all software development participants are operating in a consistent context (Boehm, 1988).

Several authors (Stachowitz and Combs, 1987; Geissman and Schultz, 1988; Culbert et al., 1988) have recommended an iterative development life cycle *similar to* the spiral life cycle model for the development of knowledge-based software. However, Boehm, (1988), the originator of the spiral life cycle model, makes a distinction between the iterative development model and the spiral model. He characterizes his spiral model as a *risk-driven* approach and iterative development model as a *code-driven* approach to the management of the development of software. The cited articles focus on the iterative (prototyping) aspect of the development of KBSs but do not directly address the risk evaluation aspect, the essential characteristic of the spiral model. Periodic risk analyses of KBSs and the evaluation of alternatives can address the issue of replacing models of human reasoning processes (e.g., heuristic search) with algorithms and other techniques when sufficient information has been learned about the problem domain through the implementation and evolution of the initial system. Friel and Mayer (1985) cite MACSYMA and R1 as examples where the initial developments were based on human reasoning, but as developers learned more about the problem domain through the implementation and evolution of the systems, most of the heuristic search aspects of the applications were replaced.

B.2.4 Transform Model

Applications developed with the waterfall model can become increasingly hard to modify when code is repeatedly enhanced (i.e., updated) and optimized for performance. It is also difficult to verify that the source code matches the requirements description and the behavioral specification of the waterfall model. The transform model, also referred to as the *operational model*, responds to these problems with an automation-based, specification-driven model as shown in Figure B-3. From the earliest stages of the life cycle the product is an executable abstraction of how the software is supposed to work. Modifications are made to the specifications, bypassing direct modification of code that has become poorly structured through repeated updates (Balzer et al., 1983). It also bypasses the intermediate design, code, and test activities. Where a compiler exists to move the specifications this close to the code, much of the work of verification becomes a one-time effort of insuring the quality of the compiler.

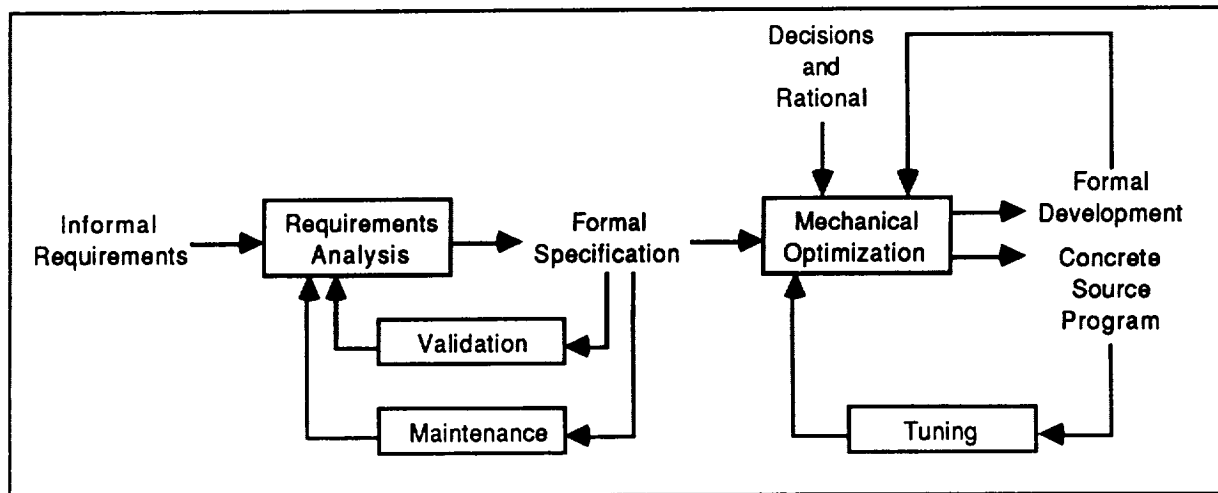


Figure B-3. Transform Life Cycle Model
(Source: Balzer et al., 1983)

This model is limited in applicability because of its requirement for a software capability to automatically convert a formal specification of a software product into a program satisfying the specification (Boehm, 1988). Currently the transform model supports spreadsheets and fourth generation language ap-

plications. However, research systems that are exploring the application of AI techniques to the knowledge-intensive activities to automate major portions of a transformational implementation may change the future of software engineering. At the present time, IBM's CASE/MVS project is applying this life cycle model to operating system development. The long-term goal is to implement a full-scale KBS, and significantly automate the process of producing tested MVS/XA product code as a result (Symonds, 1988).

Applying the transform model to the development of advanced automated software is still in the research stage. For example, Neches et al. (1985) have proposed an approach to the development of KBSs, called Explainable Expert Systems (EES), which provides an extended explanation capability and an automatic program writer to derive code from more abstract specifications thereby simplifying the maintenance process. The system developers provide a knowledge base containing descriptive knowledge of how the domain works, and abstract problem-solving methods that apply to classes of problems.

APPENDIX C

KEY CONTACTS

This section contains the names and contact information for persons that were interviewed during this study. Time and other constraints did not allow us to interview personnel from every facility discussed in this paper; information on other facilities was obtained from existing documentation and discussions with MITRE personnel familiar with the facilities.

Funding Organization

Strategic Plans and Programs, Code ST
Office of Space Station
600 Independence Ave., S.W.
Washington, D.C. 20546

Gregory E. Swietek
Manager, Advanced Automation and Robotics
(202) 453-2869
NASAMail: GESwietek

Managing Organization

Intelligent Systems Branch
Engineering Directorate
Lyndon B. Johnson Space Center
Houston, TX 77058

Kathleen J. Healey
Chief, Intelligent Systems Branch
Mail Stop: EF5
(713) 483-4776
NASAMail: KHealey/JSC/NASA

Kenneth R. Crouse
Mail Stop: EF5
(713) 483-2040
NASAMail: KCrouse/JSC/NASA

Authors

The MITRE Corporation
1120 NASA Road 1 Suite 600
Houston, TX 77058
(713) 333-5020

Steven E. Bayer
NASAMail: SBayer/JSC/NASA

Richard A. Harris
Lois W. Morgan
James F. Spitzer

JSC Data Management System Test Bed

Avionics System Division
Engineering Directorate
Lyndon B. Johnson Space Center
Houston, TX 77058

Michael M. Thomas
Mail Stop: EH431
(713) 483-8368
NASAMail: MMThomas/JSC/NASA

IBM Data Management System Test Bed

International Business Machines Corporation
3700 Bay Area Blvd.
Houston, TX 77058

Robert Hasbrouk
(713) 282-7801

JSC Operations Management System Node

Engineering Directorate
Lyndon B. Johnson Space Center
Houston, TX 77058

Max D. Holley
Space Station Engineering Office
Co-Chair OMS Integration Working Group
Special Assistant to Director, Engineering
Mail Stop: EZ
(713) 483-0404
NASAMail: MaxHolley/JSC/NASA

Allen E. Brandli
Avionics Systems Division
Co-Chair OMS Integration Working Group
Mail Stop: EH3
(713) 483-8238
NASAMail: ABrandli/JSC/NASA

PRECEDING PAGE BLANK NOT FILMED

JSC Thermal Test Bed

Crew and Thermal Systems Division
Engineering Directorate
Lyndon B. Johnson Space Center
Houston, TX 77058

Jeffrey S. Dominick
Mail Stop: EC2
(483) 525-9132
NASAMail: JDominick/JSC/NASA

ARC Thermal Test Bed/OAST TEXSYS

Systems Autonomy Demonstration Office
Ames Research Center
Moffett Field, CA 94035

Brian J. Glass
Mail Stop: 244-18
(415) 694-6525
e-mail: glass@pluto.arc.nasa.gov

LeRC Automated Electrical Power System Test

Space Station Electrical Systems Division
Space Station Systems
Lewis Research Center
21000 Brookpark Road
Cleveland, OH 44135

Jim Dolce
(216) 433-8052
NASAMail: JDolce/LERC/NASA

LeRC/OAST PMACS

Aerospace Technology Division
Lewis Research Center
21000 Brookpark Road
Cleveland, OH 44135

Edward Petrik
(216) 433-6754
NASAMail: EPetrik/LERC/NASA

MSFC Power Management and Distribution
System Test Bed

Electrical Division
Electrical Power Branch
George C. Marshall Space Flight Center
Alabama 35812

Bryan Walls
MailStop EB12
(205) 544-3311
NASAMail: BWalls/MSFC/NASA
e-mail: walls%ssl.span@fedex.msfc.nasa.gov

JSC Generic Electrical Power Distribution and
Control Test Bed

Avionic Systems Division
Engineering Directorate
Lyndon B. Johnson Space Center
Houston, TX 77058

Roberto M. Egusquiza
Mail Stop: EH3
(713) 483-8284
NASAMail: REgusquiza/JSC/NASA

ARC Advanced Architecture Test Bed

Information Systems Division
Ames Research Center
Moffett Field, CA 94035

Henry J. Lum
Chief, Information Systems Division
(415) 694-6544
NASAMail: HLum/ARC.R/ARC/NASA

Terry L. Grant
(415) 694-6526
NASAMail: TGrant/ARC.R/ARC/NASA

INCO Expert System Project

Systems Division
Mission Operations Directorate
Lyndon B. Johnson Space Center
Houston, TX 77058

John F. Muratore
Mail Stop: DF24
(713) 483-0796
NASAMail: JMuratore/JSC/NASA

Transition Flight Control Room

David Barker
Mail Stop: JSC FACC/B2B
(713) 335-6530

Space Station Program Office

Del W. Weathers
(703) 487-7248
NASAMail: DWeathers/SSE/SS/SSPO/NASA

Multi-System Integration Facility

Space Craft Software Division
Mission Support Directorate
Lyndon B. Johnson Space Center
Houston, TX 77058

Janet W. Bell
Mail Stop: FR431
(713) 483-5295
NASAMail: JBell/JSC/NASA

LIST OF REFERENCES

- Anderson, J. A. and E. Rosenfeld, eds. (1988) *Neurocomputing: Foundations of Research*, Cambridge, MA: The MIT Press.
- ARC (November 1987^a) *Systems Autonomy Technology Program (SATP) Plan*, Ames Research Center, Moffett Field, CA: NASA.
- ARC (November 1987^b) *Systems Autonomy Technology Program Plan, Executive Summary*, Ames Research Center, Moffett Field, CA: NASA.
- Balzer, R., T. E. Cheatham, Jr., and C. Green (November 1983), "Software Technology in the 1990's: Using a New Paradigm," *IEEE Computer*, pp. 39-45.
- Barr, A. and E. A. Feigenbaum, (1981) *The Handbook of Artificial Intelligence, Part I*, Stanford, CA: HeurisTech Press.
- Bochsler, D. C. and M. Goodwin (June 1986^a) "Software Engineering Techniques Used to Develop an Expert System for Automating Space Vehicle Rendezvous," *Proceedings of the Second Annual Workshop on Robotics and Expert Systems*, Instrument Society of America: Research Triangle Park, NC, pp. 87-95.
- Bochsler, D. C. and M. Goodwin (November 1986^b) "A Software Engineering Approach to Expert System Design and Verification," *Proceedings of the Conference on Artificial Intelligence for Space Applications*, Huntsville, Alabama, pp. 47-60.
- Boehm, B. W. (May 1988) "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, pp. 61-72.
- Boeing (November 1987) *Space Station Technology Development Mission Requirements Definition for Advanced Automation and Robotics, Final Report*, DRD-SE1299T, Boeing Aerospace Company.
- Citrenbaum, R. L. and J. R. Geissman (April 1986) "A Practical Cost-Conscious Expert System Development Methodology," *Proceedings of AI-86: Artificial Intelligence and Advanced Computer Technology Conference*, Long Beach, CA.
- Citrenbaum, R. L., J. R. Geissman, and R. Schultz (September 1987) "Selecting a Shell," *AI Expert*, pp. 30-39.
- Cottrell, G. W., P. Munro, and D. Zipser (February 1987) *Image Compression by Back Propagation: An Example of Extensional Programming*, Report No. 8702, Institute for Cognitive Science.
- Culbert, C., Gary Riley, and Robert T. Savely (1988) *An Expert System Development Methodology Which Supports Verification and Validation*, paper submitted to the Fourth IEEE Conference on Artificial Intelligence Applications.
- Dolce, J. L. and K. A. Faymon, (August 1987) "A Systems Engineering Approach to Automated Failure Cause Diagnosis in Space Power Systems," *22nd Intersociety Energy Conversion Engineering Conference, IECEC*.

Dolce, J. L. and K. A. Faymon (November 1986^a) "Automating the U.S. Space Station's Electrical Power System," *Optical Engineering*, Vol. 25, No. 11.

Dolce, J. (June 1987^b) *A Proposal to Investigate Space Station Power System Autonomous Control*, Submitted to Space Station Transition Definition Program in support of POP 87-2.

Dolce, J. (August 1987^c) "An Integrated Approach to Space Station Power System Autonomous Control," *22nd Intersociety Energy Conversion Engineering Conference, IECEC*.

Ecklecamp, R. and K. Reiley, eds. (June 1988) *OMS Evolution Plan Rough Draft*, NASA.

Elman, J. L. and D. Zipser (February 1987) *Understanding the Hidden Structure of Speech*, Report No. 8701, Institute for Cognitive Science.

Freeman, K. A., et al. (July 1988) *Concurrent Development of Fault Management hardware and software in the SSM/PMAD*, Proc. 23rd Intersociety Energy Conservation and Engineering Conference, p. 307-312.

Friedland, P. et al. (May 1988) *Space Station Advanced Automation Study: Final Report*, NASA Office of Space Station, Houston, TX: NASA.

Friel, P. G. and R. J. Mayer (1985) "Analysis, Engineering, and Construction Methodologies for Expert System Application Development," *Proceedings of ROBEXS '85*, Instrument Society of America, pp.1-17.

Fukushima, K. (1986) "A Neural Network Model for Selective Attention in Visual Pattern Recognition," *Biol. Cybern.*, Vol. 55, pp. 5-15

Geissman, J. R. and R. D. Schultz (February 1988) "Verification & Validation of Expert Systems," *AI Expert*, pp.26-33.

Gevarter, W. B. (May 1987) "The Nature and Evaluation of Commercial Expert System Building Tools," *Computer*, pp. 24-41.

Golden, R. M. (1986) "A Developmental Neural Model of Visual Word Perception," *Cognitive Science*, Vol. 10, pp. 241-276

Gruman, G., ed. (March 1988) "SDI Office Awards Simulation Facility Contract," *IEEE Software*.

Hayes-Roth, F. et al. (1983) *Bulding Expert Systems*, Reading, MA: Addison-Wesley Publishing Company, Inc:

Jacob, R. J. K. and J. N. Froscher, *Software Engineering for Rule-based Systems*, pp. 185-189.

Jorgensen, C. C. (1987) "Neural Network Representation of Sensor Graphs in Autonomous Robot Path Planning," *Proceedings of the 1987 AAAI Conference*, Seattle, WA, pp. IV-507-IV-515.

JSC (December 1987) *TAVERNS Concept Document*, JSC-22795, Engineering Directorate, Avionics Systems Division, Johnson Space Center, Houston, TX: NASA.

JSC (January 1988^a) *Proposal for Implementation and Contractor Participation in the JSC End-to-End Test Capability*, Space Station Program Office, Draft 2.0, JSC-22812, Johnson Space Center, Houston, TX: NASA.

JSC (January 1988^b) *Space Station Program Multisystem Integration Facility Concept Document*, Preliminary Review Version, JSC-32044, Johnson Space Center, Houston, TX: NASA.

JSC (May 1988^c) *Space Station Program Multisystem Integration Facility System Requirements Specification*, Version 0.1, Johnson Space Center, Houston, TX: NASA.

JSC (August 1988^d) *Space Station Control Center Level A Requirements*, Preliminary Version, Systems Development Division, Mission Support Directorate, JSC-32069, Johnson Space Center, Houston, TX: NASA.

JSC (August, 1988^e) *Space Station Control Center Test Bed Planning Requirements Document*, Systems Development Division, Mission Support Directorate, Johnson Space Center, Houston, TX: NASA.

Kandel, A. (1982) *Fuzzy Techniques in Pattern Recognition*, New York: John Wiley & Sons.

Kelly, C. M. (January 1988^a) "Automated Space Station Procedure Execution," *AIAA 26th Aerospace Sciences Meeting*, Reno, Nevada.

Kelly, C. M. (July 1988^b) *Conceptual Definition for a Flight Data File Automated Control and Tracking System*, MTR-88D0017, Houston, TX: The MITRE Corporation.

Kelly, C. M. (July 1988^c) *Space Station DMS Testbed Integration Effort*, Houston, TX: The MITRE Corporation.

Kish, J. A., J. L. Dolce, and D. J. Weeks (November 1988) *Space Station Power System Autonomy Demonstration*, To be presented: Advances in Intelligent Robotic Systems, SPIE's Cambridge Symposium.

Koch, C., J. Marroquin, and A. Yuille (June 1986) "Analog 'Neuronal' Networks in Early Vision," *Proc. Natl. Acad. Sci. USA*, Vol. 83, pp. 4263-4267.

The LANES User Manual (May 1987) Intelligent Systems Technology Branch, Ames Research Center, Moffett Field, CA: NASA.

LEMSCO (April 1988) *Data Management System (DMS) Test Bed User's Guide*, LEMSCO 24992, Lockheed Engineering and Management Services Company, Inc .

Lippmann, R. P. (April 1987) "An Introduction to Computing with Neural Nets," *IEEE ASAP Magazine* .

LMSC (September 1988^a) *DRLI 72 RID Status Report*, LMSC-F255472, Lockheed Missiles and Space Co., Inc.

LMSC (September 1988^b) *Functional Requirements Specification, Space Station Software Support Environment*, DRL #56, Rev. 1.0, LMSC-F255456, Lockheed Missiles and Space Co., Inc.

- Lollar, L. F. and D. J. Weeks (July, 1988) *The Autonomously Managed Power Systems Laboratory*, Proc. 23rd Intersociety Energy Conservation and Engineering Conference, p. 307-312.
- Lum, H. (January 1988) *Spaceborne VHSIC Multiprocessor System (SVMS)* Systems Autonomy Technology Program, Systems Architecture and Integration Element.
- Maiers, J. and Y. S. Sherif (January/February 1985) "Applications of Fuzzy Set Theory," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-15, No. 1, pp. 175-187.
- Mamdani, E. H. and B. R. Gaines (1981) *Fuzzy Reasoning and its Applications*, New York: Academic Press.
- Marsh, C. A. (March 1988) *The ISA Expert System: A Prototype System for Failure Diagnosis on the Space Station*, MTR-88D0013, Houston, TX: The MITRE Corporation.
- McCracken, D. D. and M. A. Jackson (April 1982) "Life-Cycle Concept Considered Harmful," ACM Software Engineering Notes, pp.29-32.
- MDC (April 2, 1988^a) *DMS Kits Concept Document*, Work Package 2, MDC H4073, McDonnell-Douglas Corporation.
- MDC (June 1988^b) *WP-2 Master Verification Plan* (DR TV-01) Work Package 2, MDC H4091, McDonnell-Douglas Corporation.
- Minsky, M. and S. Papert (1969) *Perceptrons*, Cambridge, MA: MIT Press.
- Neches, R., W. R. Swartout, and J. D. Moore (November 1985) Enhanced Maintenance and Explanation of Expert Systems Through Explicit Models of Their Development, *IEEE Transactions on Software Engineering*, pp. 1337-1351.
- Nii, H. P. (August 1986^a) "Blackboard Systems: Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective, Part Two," *The AI Magazine*.
- Nii, H. P. (Summer 1986^b) "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, Part One," *The AI Magazine*.
- Oyster, J. M. and J. Skrzypek (1987) "Computing Shape with Neural Networks," *IEEE ICNN*.
- Oyster, J. M., W. Broadwell, and F. Vicuna (January 1986) *Associative Network Applications to Robot Vision*, Report No. 320-2777, IBM Los Angeles Scientific Center.
- Ramamoorthy, C. V., S. Sheckhar, and V. Garg (January 1987), "Software Development Support for AI Programs," *IEEE Computer*, pp. 30-40.
- Richardson, K. and C. Wong (1987) *Knowledge Based System Verification and Validation as Related to Automation of Space Station Subsystems: Rationale for a Knowledge Based System Lifecycle*, NASA/Ames Research Center, Moffett Field, CA: NASA.
- Rothenberg, J. et al. (July 1987) *Evaluating Expert System Tools: A Framework and Methodology*, Santa Monica, CA: The Rand Corporation, .

Rumelhart, D. E. and J. L. McClelland (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, "Volume 1: Foundations," Cambridge, MA: The MIT Press.

Sayeh, M. R. and J. Han (1987) "Pattern Recognition Using a Neural Network," *SPIE Conf. on Adv. in Robotics Systems, Intelligent Robotics, and Computer Vision*, pp. 848-845.

Sejnowski, T. J. and C. R. Rosenberg (1986) *NETalk: A Parallel Network that Learns to Read Aloud*, Tech. Report JHU/EECS-86/01, The Johns Hopkins University, Electrical Engineering and Computer Science.

SSP (February 1988^a) *Architectural Control Document, Communications and Tracking System*, SSP-32060, Space Station Program Office, Revision A, NASA.

SSP (February 1988^b) *Architectural Control Document, Data Management System*, SSP-30261, Space Station Program Office, Revision B, NASA.

SSP (February 1988^c) *Architectural Control Document, Electrical Power System*, SSP-30263, Space Station Program Office, Revision B, NASA.

SSP (February 1988^c) *Architectural Control Document, Guidance Navigation and Control System*, SSP-32059, Space Station Program Office, Revision B, NASA.

SSP (February 1988^d) *Space Station Program Master Verification Requirements, Volume 1: Master Verification Requirements*, SSP-30467, Space Station Program Office, Draft, NASA.

SSP (April 1988^e) *Draft of SSP Verification Plan and Policies*, Draft, NASA.

SSP (July 1988^f) *Architectural Control Document, Thermal Control System*, SSP-30258, Space Station Program Office, Revision C, NASA.

SSP (July 1988^g) *SSP Information Systems Concept Document*, Draft, NASA.

Stachowitz, R. A. and J. B. Combs (January 1987) "Validation of Expert Systems," *Proceedings Hawaii International Conference on Systems Sciences*, Kona, Hawaii.

Symonds, A. J. (March 1988) "Creating a Software-Engineering Knowledge Base," *IEEE Software*, pp. 50-56.

ATAC (March 1985) *Advancing Automation and Robotics Technology for the Space Station and for the U.S. Economy, Volume II - Technical Report*, TM-87566, NASA Advanced Technology Advisory Committee (ATAC) NASA.

Turban, E. (May 1988) "Review of Expert Systems Technology," *IEEE Transactions on Engineering Management*, pp. 71-81.

Ulmer, J. W. (June 1988) *OMS Integrated Test Bed Demo 3 Test Plan*, Houston, TX: TRW Houston System Services.

Waechter, D. J. and Walling, S. S. (February 1988) *Multipurpose Interface Device 1553 Simulation Sub-Unit*, IBM-87-C72-003, IBM.

Weeks, D. J. (June 1988) "Artificial Intelligence Approaches in Space Power Systems Automation at Marshall Space Flight Center," *Proc. of the First International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, Vol. 1, University of Tennessee Space Institute, Tullahoma, TN, pp 361-366.

Wong, C. M. et al. (1988) *Cooperating Expert Systems for Space Station: Power/Thermal Subsystem Testbeds..*

Zadeh, L. A. (1965^a) "Fuzzy Sets," *Information and Control*, Vol. 8, pp. 338-353.

Zadeh, L. A. (1983^b) "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems," *Fuzzy Sets and Systems*, Vol. 11, pp. 199-227.

Zadeh, L. A. (October 1983^c) "Commonsense Knowledge Representation Based on Fuzzy Logic" *Computer*, Vol. 16, No. 10, pp. 61-65.

Zadeh, L. A. (April 1988^d) "Fuzzy Logic," *Computer*, Vol. 21, No. 4, pp. 83-93.

GLOSSARY

A&R	Automation and Robotics
ADC	Analog-to-Digital Card
ADI	Applied Dynamics International
ADS	Ancillary Data Services
ADS	Automated Document System
AGE	Aerospace Ground Equipment
AI	Artificial Intelligence
Aiestwg	Artificial Intelligence, Expert Systems, and Technology Group
AMPSLAB	Autonomously Managed Power Systems Laboratory
APAE	Attached Payload Accommodations Equipment
APEX	Automated Power Expert
ARC	Ames Research Center
ARCHANGEL	Autonomous Real-time Control of Hierarchical Agents and Networks at the Global Executive Level
ART	Automated Reasoning Tool
AT	Advanced Technology
ATAC	Advanced Technology Advisory Committee
ATCS	Active Thermal Control System
BIU	Bus Interface Units
C&T	Communications and Tracking
CCZ	Command and Control Zone
CLOS	Common Lisp Object System
CMS	Control and Monitoring Subsystem
CMZG	Control Movement Gyro
COTS	Commercial off-the-shelf
CRT	Cathode Ray Tube
CY	Calendar Year
D&C	Displays and Control
DACS	Data Acquisition and Control System
DADS	Data Acquisition and Distribution Services
DC	Direct Current
DEC	Digital Equipment Corporation
DMA	Direct Memory Access
DMS	Data Management System
DPNS	Distributed Processing Network Simulator
ECLSS	Environmental Control and Life Support System
EES	Explainable Expert Systems
EPS	Electrical Power System
ESTEC	European Space Technology Center
ETC	End-to-End Test Capability
EVA	Extra-Vehicular Activity
FDDI	Fiber Distributed Data Interface
FDF	Flight Data File
FDIR	Fault Detection Isolation and Recovery
FEU	Functional Equivalent Unit

FLIPS	Fuzzy Logical Inferences Per Second
FRAMES	Fault Recovery and Management Expert System
FTS	File Transfer Services
FY	Fiscal Year
Gbyte	Gigabyte
GEPDC	Generic Electric Power Distribution and Control
GFLOPS	Billions of Floating Point Operations per Second
GN&C	Guidance, Navigation, and Control
GPS	Global Positioning System
GSFC	Goddard Space Flight Center
HAE	HCI Ada Executive
HCI	Human Computer Interface
HIP	Human Interface to Power
HITEX	Human Interface to TEXSYS
HW	Hardware
IBM	International Business Machines
IESP	INCO Expert System Project
IMU	Inertial Mass Unit
IIM	Integrated Inference Machines
INCO	Integrated Communications Officer
IO	Input/Output
IOC	Initial Operations Capability
IRR	Initial Requirement Review
ISA	Integrated Status Assessment
ISB	Intelligent Systems Branch
IT&V	Independent Test & Verification
IV&V	Independent validation and verification
JSC	Johnson Space Center
KBS	Knowledge-Based System
KEE	Knowledge Engineering Environment
kHz	Kilohertz
KVA	Kilovolts Alternating Current
kW	Kilowatt
LAN	Local Area Network
LANES	Local Area Network Extensible Simulator
LeRC	Lewis Research Center
LES	Loads Enable Scheduler
LLP	Lowest Level Processors
LMSC	Lockheed Missiles and Science Corporation
LPLMS	Loads Priority List Management System
Mbps	Megabits per second
Mbyte	Megabyte
MBS	Megabyte per second
MBSA	Main Bus Switching Assembly
MCC	Mission Control Center
MCCU	MCC Upgrade

MDM	Multiplexer-Demultiplexer
MFLOPS	Millions of Floating Point Operations per Second
MID	Multipurpose Interface Device
MIL	Military
MIPS	Millions of Instructions per second
MIS	Management Information Systems
MOD	Mission Operations Directorate
MPAC	Multi-Purpose Applications Console
MPAD	Mission Planning and Analysis Division
MSID	Measurement Simulation Identification
MSC	Mobile Servicing Center
MSD	Mission Support Directorate
MSFC	Marshall Space Flight Center
MSIF	Multi-System Integration Facility
MTK	Model Tool Kit
NASA	National Aeronautics and Space Administration
NIE	Network Interface Element
NIU	Network Interface Units
NOS	Network Operating System
NOSLIB	Network Operating System Library
NSE	Network Service Element
NSTS	National Space Transportation System
OASIS	Operations and Science Instrument Support
OAST	Office of Aeronautics and Space Technology
OBCO	On Board Check Out
OMA	Operations Management Application
OMGA	Operations Management Ground Application
OMS	Operations Management System
OMV	Orbital Maneuvering Vehicle
ORU	Orbital Replaceable Unit
OSI	Open Systems Interconnect
OSS	Office of Space Station
PC	Personal Computer
PDCU	Power Distribution Control Unit
PDR	Preliminary Design Review
PDRD	Program Definition and Requirements Document
PI	Procedures Interpreter
PLS	Payload Simulator
PMACS	Power Management and Control System
PMAD	Power Management and Distribution
PMC	Power Management Controller
PPD	Projection Plotting Display
PSAD	Power System Autonomy Demonstration
PSCN	Program Support Communications Network
PTAD	Power-Thermal Autonomy Demonstration
PTCS	Passive Thermal Control System
RAD	Read Only Memory
RCS	Reaction Control System
REX	Rendezvous Expert System

RF	Radio Frequency
SADP	Systems Autonomy Demonstration Project
SATP	Systems Autonomy Technology Program
SDD	Systems Development Division
SDI	Strategic Defense Initiative
SDP	Standard Data Processor
SDP	Standard Data Processors
SDR	System Design Review
SFDF	Station Flight Data File
SIB	Simulation Interface Buffer
SIF	System Integration Facility
SPF	Software Production Facilities
SRD	System Requirements Documentation
SSCC	Space Station Control Center
SSE	Software Support Environment
SSEDF	Software Support Environment Development Facility
SSFP	Space Station Freedom Program
SSFPO	Space Station Freedom Program Office
SSIS	Space Station Information System
SSM	Space Station Module
SSFPE	Space Station Freedom Program Elements
SSTF	Space Station Training Facility
SUT	System Under Test
SVMS	Spaceborne VHSIC Multiprocessor System
TAVERNS	Test And VERification of Remote Networked Systems
TCAS	Test Control and Simulation Environment
TCP/IP	Transmission Control Protocol/Internet Protocol
TCS	Thermal Control System
TDAS	TEXSYS Data Acquisition System
TDRSS	Tracking and Data Relay Satellite System
TEXSYS	Thermal Expert System
TFCR	Transition Flight Control Room
TI	Texas Instruments
UIL	User Interface Language
UIMS	User Interface Management System
UUT	Under Test
V&V	Verification and Validation
VAC	Volts Alternating Current
VAD	Volts Direct Current
VDB	Verification Data Base
VHSIC	Very High Speed Integrated Circuit
WEX	Windowing Executive
WP	Work Package
WPL	Workstation Prototype Lab
WS	Work Station
XTK	Executive Tool Kit

DISTRIBUTION LIST

A10

E. L. Key
C. A. Zraket

D11

B. M. Horowitz

D12

J. J. Feamsides
S. W. Gouse
C. C. Grandy

D22

Technical Report Center (2)

W107

Records Resources (2)

W120

A. H. Ghovanlou
E. S. Herndon
R. E. Smiley

W121

G. B. Allison
R. Khare
K. H. Miller
A. J. Nickelson
J. H. Noles
A. O. Sipes (PASA)
M. G. Walker
A. D. Zeitlin

W122

S. N. Goldstein (NAHQ)
E. E. Hill (NAHQ)
R. B. Hoffman (REST)

R. W. Hopkins (REST)
J. V. Pietras (Gren)
C. R. Somerlock (NAHQ)
J. N. Williams (Gren)
F. L. Willingham (REST)

W123

R. D. Gilreath
K. P. Hennigan
H. C. Hillman
S. I. Linde
A. T. Lovelace
T. R. Mitchell
F. M. Richards
J. K. Richardson

W124

S. E. Bayer (10)
S. A. Bell
D. M. Erb
I. J. Feig
P. J. Gregor
R. A. Harris
R. M. Jackson
L. W. Morgan
A. N. Rasmussen
J. C. Reynolds
J. F. Spitzer

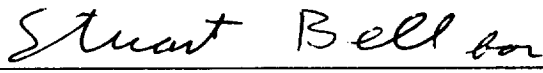
W125

J. S. Brown
P. M. Brown
R. C. Mitchell
R. M. Myers
W. B. Wood
R. W. Zears

NASA

Under separate distribution list (100)

Approved for Project Distribution


E. S. Herndon

